2017

Practical Functional Encryption Techniques and Their Applications

Shiwei Zhang
University of Wollongong

UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Practical Functional Encryption Techniques and Their Applications

Shiwei Zhang

Supervisor:
Prof. Yi Mu
Co-supervisor:
Dr. Guomin Yang

This thesis is presented as required for the conferral of the degree:

Doctor of Philosophy

The University of Wollongong
School of School of Computing and Information Technology

May 2017

Declaration

I, Shiwei Zhang, declare that this thesis submitted in fulfilment of the requirements for the conferral of the degree Doctor of Philosophy, from the University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualifications at any other academic institution.

Shiwei Zhang
May 5, 2017

Abstract

Confidentiality can be usually achieved by conventional encryption, which hides sensitive data from unauthorised parties. In such encryption schemes, the decryption is all or nothing. Although it ensures the confidentiality of the encrypted data, it does not allow computing over files and sharing. Differing from the conventional encryption, functional encryption allows the decryption key holder to learn a function of the encrypted data and nothing else. Therefore, functional encryption makes computing over encrypted data and sharing of encrypted data in different levels possible.

In this thesis, we investigate functional encryption in terms of functionality, security, and efficiency. To ensure the constructed schemes are practical in real applications, we focus on functional encryption for practical functionalities, such as equality tests, inequality tests, and inner product evaluation, which are the major functionalities that can be applied in privacy-preserved data search and privacy-preserved data sharing. Precisely, functional encryption for equality tests and inequality tests can be applied in searchable encryption while the functional encryption for inner product evaluation can be applied to achieve levelled data sharing. These applications of functional encryption are very useful and essential in cloud computing security.

Public-key Encryption with Keyword Search (PEKS) is a searchable encryption, which can be viewed as a functional encryption for equality tests. In PEKS, a keyword is encrypted and can be searched by a single user. Based on PEKS, we propose two new cryptographic primitives named Threshold Broadcast Encryption with Keyword Search (TBEKS) and Linear Encryption with Keyword Search (LEKS). In TBEKS, the keyword is encrypted for a group of users as a single ciphertext with a threshold value. To achieve a higher level of security, the data search process can be carried out by the server only if the number of authorised users reaches the threshold value defined in the ciphertext. Thus a single malicious user cannot carry out a search operation if the stored data is highly sensitive. Besides, the data in the storage server may be encrypted using different encryption mechanisms, such as Identity-Based Encryption (IBE) and Attribute-Based Encryption (ABE). With LEKS, we allow conversions from various encryption schemes to searchable

encryption schemes, keeping the same access control mechanism. Notably, we convert a Key-Policy Attribute-Based Encryption (KP-ABE) scheme to a Key-Policy Attribute-Based Encryption with Keyword Search (KP-ABKS) scheme, enabling the data to be accessed and searched using fine-grained access control policies.

For data sharing, we propose a practical Functional Encryption for Inner Products (FE-IP) scheme to control the revelation of the encrypted data. In FE-IP, the secret key holder with a vector $\vec{x}$ can learn the value of $\vec{x} \cdot \vec{y}$ from the encrypted vector $\vec{y}$ but not the vector $\vec{y}$ itself. Such a functionality can be applied to descriptive statistics such as computing weighted sums and weighted means. Furthermore, we generalise the functionality of inner products to matrix products, and upgrade functional encryption to the more advanced Hierarchical Functional Encryption (HFE). In HFE, the secret key holder can further generate a new key with a narrower portion of its decryption ability. As a result, we propose a Hierarchical Functional Encryption for Linear Transformations (HFE-LT) scheme. In addition, we extend the definition of HFE to extended Hierarchical Functional Encryption (eHFE) and the construction of HFE-LT to extended Hierarchical Functional Encryption for Linear Transformations (eHFE-LT) so that multiple secret key holders can work together to generate a new key with a narrower portion of the union of their decryption abilities.

Notably, all schemes in this thesis are formally proven secure in their respective security models. In order to prove the security of LEKS, we introduce a new problem family named Decisional Bilinear (P, f)-Diffie-Hellman ((P, f)-DBDH) problem family and a derived problem named Decisional ℓ-Combined Bilinear Diffie-Hellman (ℓ-DCBDH) problem and analyse the difficulty of these problems in the generic group model. For FE-IP, we consider the strongest security model, namely Indistinguishability under adaptive Chosen Ciphertext Attacks (IND-CCA) for functional encryption, and prove the IND-CCA security of our FE-IP scheme. For all the generic constructions in this thesis, we propose practical concrete schemes instantiated from various assumptions.

Acknowledgments

I would like to express my heartfelt thanks to my supervisors, Professor Yi Mu, and Dr. Guomin Yang, for their careful guidance, helpful suggestions, and patience. I appreciate their invaluable encouragement when I was completely frustrated, especially at the beginning of my PhD candidacy. Without their support, this thesis would not exist.

I wish to thank my colleagues during my PhD candidacy for their kindness and support. The non-exhaustive list of whom includes: *Peng Jiang, Xiaofen Wang, Fuchun Guo, Nan Li, Rongmao Chen, Zhenfei Zhang, Kefeng Wang, Hui Cui, Yinhao Jiang, Yangguang Tian, Shengmin Xu, Weiwei Liu, Jiannan Wei, Jianchang Lai, Zhongyuan Yao, Ge Wu, Tong Wu, Fatemeh Rezaeibagha, and Tran Viet Xuan Phuong.* I would also like to thank my friend *Yingzhi Gou* for his insights on knowledge of computer science, as well as his organising of social activities.

It is my honour to be a member of Centre for Computer and Information Security Research (CCISR), School of Computing and Information Technology, University of Wollongong. I would like to thank University of Wollongong for providing me the scholarships of University Postgraduate Award (UPA) and International Postgraduate Tuition Award (IPTA) and supporting my PhD research.

I dedicate this thesis to the memory of my mother *Hong Mei*, whose role in my life was, and remains, immense. Last but not least, I am sincerely grateful to my father *Qingyuan Zhang* and my family for their love and encouragement, and for everything.

Shiwei Zhang
March 2017
Wollongong, NSW, Australia

Publications

This thesis is based on the following published or presented papers, which were finished when I was in pursuit of the PhD degree in University of Wollongong.

1. Shiwei Zhang, Yi Mu, and Guomin Yang. Threshold broadcast encryption with keyword search. In Dongdai Lin, XiaoFeng Wang, and Moti Yung, editors, *Information Security and Cryptology: 11th International Conference, Inscrypt 2015, Beijing, China, November 1–3, 2015, Revised Selected Papers*, pages 322–337, Cham, 2016. Springer International Publishing.

2. Shiwei Zhang, Guomin Yang, and Yi Mu. Linear encryption with keyword search. In Joseph K. Liu and Ron Steinfeld, editors, *Information Security and Privacy: 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4–6, 2016, Proceedings, Part II*, pages 187–203, Cham, 2016. Springer International Publishing.

3. Shiwei Zhang, Yi Mu, and Guomin Yang. Achieving ind-cca security for functional encryption for inner products. In Kefei Chen, Dongdai Lin, and Moti Yung, editors, *Information Security and Cryptology: 12th International Conference, Inscrypt 2016, Beijing, China, November 4–6, 2016, Revised Selected Papers*, pages 119-139, Cham, 2017. Springer International Publishing.

4. Shiwei Zhang, Yi Mu, Guomin Yang, and Xiaofen Wang. Hierarchical functional encryption for linear transformations. In *Information Security and Privacy: 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3–5, 2017, Proceedings*. Springer International Publishing. (accepted on 7th April 2017)

Contents

List of Figures

List of Tables

List of Abbreviations

(P, f)-DBDH Decisional Bilinear (P, f)-Diffie-Hellman. iv, 5, 7, 14, 50, 53–55, 58, 60, 84, 129

ℓ-DCBDH Decisional ℓ-Combined Bilinear Diffie-Hellman. iv, 5, 7, 50, 53, 54, 60, 62, 72, 74, 79, 83, 84, 129

$i\mathcal{O}$ indistinguishable obfuscation. 114, 115

q-SDH q-Strong Diffie-Hellman. 53

ABE Attribute-Based Encryption. iii, 7, 21, 22, 25, 26, 51–53, 130

ABKS Attribute-Based Encryption with Keyword Search. 37, 50–53

AnonHIBE Anonymous Hierarchical Identity-Based Encryption. 37

AnonHICBE Anonymous Hierarchical Identity-Coupling Broadcast Encryption. 37

AnonIBE Anonymous Identity-Based Encryption. 25, 27–29, 37

BE Broadcast Encryption. 3, 52

BEKS Broadcast Encryption with Keyword Search. 37

CDH Computational Diffie-Hellman. 12, 13, 19, 20, 53

CNF conjunctive normal form. 28

CP-ABE Ciphertext-Policy Attribute-Based Encryption. 6, 21, 25, 26, 29, 33, 52, 88

CP-ABKS Ciphertext-Policy Attribute-Based Encryption with Keyword Search. 52

CRHF Collision Resistant Hash Function. 16, 107

DBDH Decisional Bilinear Diffie-Hellman. 5, 13, 19, 35, 38, 45–47, 49, 129

DCR Decisional Composite Residuosity. 6, 7, 14, 15, 86, 89, 90, 106, 108–111, 114–116, 124, 126, 127, 130

DDH Decisional Diffie-Hellman. 6, 7, 12–14, 19, 53, 55, 86, 88–90, 106, 107, 109–111, 114–116, 124, 127, 130

DFA Deterministic Finite Automata. 22, 88, 89

DL Discrete Logarithm. 12, 13, 19, 42, 53

DLIN Decision Linear. 13, 14, 19, 21, 53, 79

DNF disjunctive normal form. 28

DPVS Dual Pairing Vector Space. 21

eHFE extended Hierarchical Functional Encryption. iv, 6, 7, 115, 116, 123, 127, 130

eHFE-LT extended Hierarchical Functional Encryption for Linear Transformations. iv, 6, 7, 115, 123, 124, 127, 130

FE Functional Encryption. xii, 2–4, 6, 7, 20–25, 27, 29, 32, 33, 52, 86–89, 92, 99, 110, 111, 113–117, 130

FE-ET Functional Encryption for Equality Tests. 5, 27, 28, 33, 130

FE-IP Functional Encryption for Inner Products. iv, xii, 6, 7, 22, 32, 33, 86–90, 92, 99, 100, 102, 110, 114, 129, 130

FE-IT Functional Encryption for Inequality Tests. 27–29, 33

HFE Hierarchical Functional Encryption. iv, xi, 6, 7, 111–118, 123, 127, 130

HFE-LT Hierarchical Functional Encryption for Linear Transformations. iv, 6, 7, 115–118, 120, 124–127, 130

HIBE Hierarchical Identity-Based Encryption. 114–116

HPS Hash Proof System. 6, 7, 16, 17, 86, 89, 90, 93–100, 102–104, 106–111, 114–116, 118, 120, 122, 124, 126, 127, 130

HVE Hidden Vector Encryption. 25, 26, 88

IBE Identity-Based Encryption. iii, 3, 13, 20, 22, 25, 37, 52, 53, 88

IBKS Identity-Based Searchable Encryption. 37, 52, 53

IND-CCA Indistinguishability under adaptive Chosen Ciphertext Attacks. iv, xi, 3, 6, 7, 22, 86, 87, 89–94, 99, 102, 103, 110, 114, 130

IND-CKA Indistinguishability under Adaptive Chosen Keyword Attacks. xi, 5, 27, 53, 64, 66, 70, 72–74, 129

IND-CPA Indistinguishability under Chosen Plaintext Attacks. xi, 3, 6, 7, 22–25, 27, 29, 33, 86–88, 91, 93, 94, 111, 114, 115, 117, 120, 121, 123, 124, 127

IND-sCKA Indistinguishability under Selective-ID Adaptive Chosen Keyword Attacks. xi, 5, 50, 53, 54, 64–66, 75, 79, 83, 129

IND-T-CKA Indistinguishability in the threshold setting under Chosen Keyword Attacks. xi, 5, 7, 35, 38, 41, 42, 45, 47, 49, 129

IPE Inner Product Encryption. 25, 26, 32, 88

KGA Keyword Guessing Attack. 36, 37, 40, 64, 130

KGC Key Generation Centre. 20

KP-ABE Key-Policy Attribute-Based Encryption. iv, 5, 7, 21, 22, 25, 26, 50, 52, 54, 75–78, 84, 88, 129

KP-ABKS Key-Policy Attribute-Based Encryption with Keyword Search. iv, 5, 7, 50, 52, 54, 75, 78, 79, 84, 129

LEKS Linear Encryption with Keyword Search. iii, iv, 5, 7, 50, 53, 54, 63–71, 75, 77, 84, 129

LET Linear Encryption Template. 5, 7, 50, 53, 54, 66–71, 75–77, 84, 129, 130

LEW Learning With Error. 88, 114

MC Match-Concealing. 29

MIFE Multi-Input Functional Encryption. 22

MR Match-Revealing. xi, 29

NIPE Non-zero Inner Product Encryption. 26

NIZK Non-Interactive Zero Knowledge. 89, 99

PE Predicate Encryption. 7, 22, 24–28, 33, 64, 88, 113, 114

PE-IT Predicate Encryption for Inequality Tests. 27–29

PEKS Public-key Encryption with Keyword Search. iii, 3, 7, 22, 27, 36, 37, 42, 51–54, 70, 72, 73, 84, 129

PKC Public-Key Cryptography. 2

PKE Public Key Encryption. 2–5, 7, 20, 52–54, 66, 70, 84, 91, 94, 114, 120, 129

PPT probabilistic polynomial-time. 11–14, 16

s-IND-CPA Selective Security against Chosen Plaintext Attacks. 6, 86, 88, 113, 114

SMP Subset Membership Problem. 6, 14, 16, 86, 89, 103, 110, 118, 120, 121, 124

TBE Threshold Broadcast Encryption. 38, 42

TBEKS Threshold Broadcast Encryption with Keyword Search. iii, 5, 7, 35, 38–40, 42, 44, 45, 49, 129

TPEKS Threshold Public-key Encryption with Keyword Search. 37, 38

ZIPE Zero Inner Product Encryption. 26, 28, 29

Part I

Introduction to Functional Encryption

Chapter 1

Introduction

In the Information Age, hiding sensitive data is one of the fundamental problems in information security and has attracted a lot of attentions. To provide confidentiality, many cryptography tools such as encryption are created to make sensitive information out of the adversary's reach. While confidentiality and privacy are preserved, it is also beneficial to allow sharing of information to authorised parties at the same time, especially in the presence of Cloud Computing [MG11]. In addition, the efficiency of the cryptographic tools is also an important aspect in order to ensure those tools can be used in practice. In this thesis, we introduce and enhance several cryptographic primitives in the domain of Functional Encryption (FE) in terms of functionality, security, and efficiency, and demonstrate their usage in some real applications and cloud computing.

1.1 Background

For many centuries, conventional cryptography has been used to secure communications among multiple parties with secret keys shared among them. Such kind of cryptography is usually symmetric-key-based and belongs to secret-key cryptography[1]. Differing from the conventional cryptography, Diffie and Hellman [DH76] suggested new directions to Public-Key Cryptography (PKC) where users hold asymmetric key pairs of a secret key and a public key. Precisely, the public key is linked with the secret key and can be released to the entire public domain whilst the secret key is infeasible to be computed from the public key.

As in [DH76], Diffie and Hellman introduced the notions of Public Key Encryption (PKE) and Digital Signature, which is a mimic of traditional hand-written signatures in the digital world. Focusing on PKE, three directions are intensively researched in the succeeding literature, including functionality, security, and efficiency.

[1] In the literature, secret-key cryptography sometimes is also referred to as private-key cryptography

Funcitonality In a standard PKE scheme, such as RSA encryption [RSA78], ElGamal encryption [ElG85], and Cramer-Shoup encryption [CS02], there are three algorithms namely Setup for system setup and key generation, Encrypt for encryption and Decrypt for decryption. However, the standard schemes are too basic to be applied to more advanced scenarios. Therefore, various encryption schemes for different functionalities are proposed. For instances, there are Broadcast Encryption (BE) [FN94] for a set of recipients, Identity-Based Encryption (IBE) [Sha85, BF01] for users authenticated by their identities, and Public-key Encryption with Keyword Search (PEKS) [BDCOP04] for searchable encryption. After several rounds of evolution, the notion of Functional Encryption (FE) [BSW11] is introduced to capture a large class of functionalities.

Security In the early stage of cryptography research, there is only an intuitive or informal security analysis for the proposed schemes [DH76, RSA78, ElG85, Sha85]. To formally define the security, Goldwasser and Micali [GM84] suggested *semantic security*, which is equivalent to Indistinguishability under Chosen Plaintext Attacks (IND-CPA), as the standard security notion for PKE schemes. Generally speaking, a security model involves the syntax of the scheme, the goal of the adversary, and the attack model. For IND-CPA security of a PKE scheme, the goal is to distinguish two messages from a targeted ciphertext, and the attack model allows the adversary to encrypt any message. To prove the security of a scheme, a simulator embedded with a (computationally) hard problem is constructed and presented to the adversary. If the adversary follows the security model and breaks the scheme, the simulator can solve the hard problem. Since the embedded problem is hard, by contradiction, the security of the scheme is ensured under the specific security model. In some applications, the IND-CPA security is not sufficient and the security model is elevated to Indistinguishability under adaptive Chosen Ciphertext Attacks (IND-CCA)[2] [RS92], giving more power to the adversary. It is worth to note that PKE schemes with different functionalities have different syntax. Thus their security models are not compatible with that of the standard PKE. Therefore, new security models are demanded for proper security proofs for various schemes.

Efficiency In terms of practicality and usefulness, all the schemes should be efficient not only in theory but also in practice. Initially, functional encryption schemes are proposed to handle a large class of functionalities, and even generic functionalities. Although the functionality is powerful, the efficiency is a ma-

[2]The conventional name for the IND-CCA security is IND-CCA2.

jor concern when the generic scheme is applied to a specific scenario. In other words, a direct construction for a specific functionality is at least as efficient as the scheme for a function class. For instance, Garg et al.'s FE for generic circuits [GGH+13] is efficient[3] in theory but it is significantly less efficient than FE schemes [ABCP16, ABDCP15, ALS16, ZMY17] for the specific functionality of inner product evaluation, and is not suitable for practical use.

In practice, the standard PKE ensures the confidentiality of the user data but it is not capable of achieving computing over encrypted files or flexible sharing of encrypted data to authorised parties. In contrast, both scenarios are realisable by Functional Encryption. Differently, the decryption of FE learns a function of the encrypted data but nothing else while the decryption of the standard PKE is all or nothing. This special decryption ability makes FE powerful to be applied in practice. However, theoretical FE schemes [GGH+13, GKP+13] aim at general functionalities and are not practical. Therefore, practical FE schemes for specific functionalities are focused, such as data search and data sharing, which are two major functionalities that require specific cryptosystems to preserve privacy, especially in cloud computing.

Data Search A common function computed over data is search. In privacy preserved data search, the keyword is encrypted and uploaded to the data server. Then the server tests the encrypted keyword with the encrypted data, and returns the result to the user. Since the data to be searched may be encrypted under different access policies, efficient constructions of secure searchable schemes for those policies are required.

Data Sharing In the privacy preserved data storage, the user data is encrypted for a certain entity and cannot be changed. To share the data, two naive ways are sharing the secret keys and re-encrypting the data. Obviously, sharing the secret keys is not an acceptable solution because other user who gets the shared key may be able to do malicious tasks. On the other hand, re-encryption of the data for each user is costly since the data is duplicated and the storage is wasted. Therefore, the task of creating efficient privacy-preserving data sharing schemes that share data to multiple users without increasing storage cost is a challenging research problem.

1.2 Motivations and Contributions

The contributions of this thesis are summarised as follows.

[3]In theory, an algorithm is efficient if it can be computed in probabilistic polynomial time.

Data Search The basic searchable encryption scheme is introduced by Boneh et al. [BDCOP04]. In their scheme, the privacy preserving keyword search for cloud data is a one-to-one relationship that one ciphertext is only for one user. In other words, the cloud has to encrypt the same contents to multiple users as multiple ciphertexts, which is storage consuming. To tackle this problem, we introduce several searchable schemes with access control applied to the ciphertext so that the same keyword can be encrypted for and searched by multiple users as a single ciphertext.

We introduce a new notation named Threshold Broadcast Encryption with Keyword Search (TBEKS). In TBEKS, the keyword is encrypted for n users as a single ciphertext. The keyword can only be search when t of n target users work together. Furthermore, we provide a formal definition of TBEKS with a security model named Indistinguishability in the threshold setting under Chosen Keyword Attacks (IND-T-CKA). Based on Decisional Bilinear Diffie-Hellman (DBDH) assumption, we propose a practical TBEKS scheme with IND-T-CKA security.

We introduce another new notation named Linear Encryption with Keyword Search (LEKS) and define a Linear Encryption Template (LET). With LEKS, we are able to convert any PKE, which matches our LET, to a searchable encryption scheme with the same access control mechanism. Specifically, we formalise the definition of LEKS and LET. We provide two security models for LEKS, namely Indistinguishability under Selective-ID Adaptive Chosen Keyword Attacks (IND-sCKA) and Indistinguishability under Adaptive Chosen Keyword Attacks (IND-CKA). Since our conversion from LET to LEKS is semi-generic, we introduce a new problem family named Decisional Bilinear (P, f)-Diffie-Hellman ((P, f)-DBDH) problem family as a tool to enable provable security. We prove the newly introduced (P, f)-DBDH problem is computationally hard in the generic group model if the polynomial f is not dependent on P. To illustrate the feasibility of our semi-generic framework, we convert a Key-Policy Attribute-Based Encryption (KP-ABE) scheme into a Key-Policy Attribute-Based Encryption with Keyword Search (KP-ABKS) scheme, which realises searchable encryption with a fine-grained access control and is suitable for cloud computing scenarios. We prove the converted KP-ABKS scheme is IND-sCKA secure based on the computational hard Decisional ℓ-Combined Bilinear Diffie-Hellman (ℓ-DCBDH) problem derived from the (P, f)-DBDH problem.

We emphasise that TBEKS and LEKS are variants of Functional Encryption for Equality Tests (FE-ET).

Data Sharing Conventionally, the user data in the privacy preserved data storage is managed under an access policy. For instance, in Ciphertext-Policy Attribute-Based Encryption (CP-ABE), the data can be decrypted only if the attributes of the user matches the policy embedded in the ciphertext. In this thesis, we focus on another approach of data sharing. Instead of decrypting the full message from the ciphertext under a certain access control policy, we control the revelation of the encrypted message via Functional Encryption (FE) [BSW11] such that a valid user can only learn a function of the encrypted message. Other than FE for general functionalities, we focus on a practical functionality of inner product evaluation (i.e. $f(\vec{x}, \vec{y}) = \langle \vec{x}, \vec{y} \rangle = \vec{x} \cdot \vec{y}$). This functionality can be applied to descriptive statistics such as computing weighted sums and weighted means. By distributing different secret keys of $f_{\vec{x}}$ to users, the data are shared with different levels of information revelation.

In order to enhance the security notions of Selective Security against Chosen Plaintext Attacks (s-IND-CPA) [ABDCP15] and IND-CPA [ALS16, ABCP16], we define a stronger security model for FE named IND-CCA. To build an IND-CCA secure Functional Encryption for Inner Products (FE-IP), we extend the exisiting Hash Proof System (HPS) [CS02] and propose an FE-IP generically constructed from our extended HPS. We prove that our generic construction of FE-IP is IND-CCA secure under the assumption of a hard Subset Membership Problem (SMP). In addition, we propose two practical instantiations from the Decisional Diffie-Hellman (DDH) and the Decisional Composite Residuosity (DCR) assumptions.

We generalise the functionality of inner product evaluation to matrix product evaluation. Besides, we refine the definition of Hierarchical Functional Encryption (HFE) [ABG+13, BS15, CGJS15], and formalise a new primitive named Hierarchical Functional Encryption for Linear Transformations (HFE-LT). With HFE, the secret key holder is able to generate a new key with ability to reveal a narrower portion of its data share. Based on our extended HPS, we propose a generic construction of HFE-LT with proven IND-CPA security, assuming the underlying SMP is hard. We provide two practical HFE-LT schemes instantiated from the DDH and the DCR assumptions. In addition, we extend HFE and HFE-LT to extended Hierarchical Functional Encryption (eHFE) and extended Hierarchical Functional Encryption for Linear Transformations (eHFE-LT), allowing multiple function inputs in the key generation algorithm.

As an additional contribution, this thesis presents a novel encoding technique for improving the efficiency of the inequality test evaluation by reducing the number of

secret-sharing reconstruction from $O(\log n)$ as required by using the access control tree in the Attribute-Based Encryption (ABE) [GPSW06, BSW07] to $O(1)$.

1.3 Thesis Organisation

The remainder of this thesis is organised as follows.

In Chapter 2, we review the preliminaries. Starting from the basic knowledge abstract algebra, we define the notations used by this thesis. Then, we review the computational complexity theory and related complexity assumptions. After that, we review the cryptographic tools used as building blocks and used in security proof.

In Chapter 3, we explore the notion of Functional Encryption. At the beginning, we review the history of FE. Then, we review the formal definition of FE and its IND-CPA security. Furthermore, we discuss the FE schemes for practical functionalities, including Predicate Encryption (PE), equality test, inequality test, and inner product evaluation. In the discussion of inequality test, we introduce an encoding technique to improve the decryption efficiency of the PE with public index related to inequality test.

In Chapter 4, we define Threshold Broadcast Encryption with Keyword Search and its IND-T-CKA security model. Then we propose our TBEKS scheme, and prove it is IND-T-CKA secure.

In Chapter 5, we define the (P, f)-DBDH problem family and derive the ℓ-DCBDH problem. We prove the hardness of the problems in terms of computational complexity and propose our LEKS conversion from LET with definition and security model formally defined. Then, we convert a PKE scheme to a PEKS scheme, and prove its security. We also convert a KP-ABE scheme to a KP-ABKS scheme with security proven.

In Chapter 6, we present a precise definition of Functional Encryption for Inner Products. We enhance the security model for FE to IND-CCA and extend the existing HPS with new properties. After that, we propose a FE-IP scheme generically constructed from the extended HPS. Furthermore, we prove our scheme is IND-CCA secure. Finally, we instantiate our FE-IP scheme to two concrete schemes from the DDH and the DCR assumptions.

In Chapter 7, we refine Hierarchical Functional Encryption and define Hierarchical Functional Encryption for Linear Transformations. We propose a HFE-LT scheme generically constructed from the extended HPS. After proving the security of our newly proposed scheme, we extend the notion of HFE to eHFE and the notion of HFE-LT to eHFE-LT. Finally, we present two concrete HFE-LT schemes instantiated from the DDH and the DCR assumptions.

In Chapter 8, the conclusion of this thesis is presented.

Chapter 2

Preliminaries

In this chapter, we cover the definitions and the notations used throughout this thesis, including abstract algebra, computational complexity and assumptions, and cryptographic tools. For the detailed basic cryptography theory, we refer readers to the books [Gol01, Gol08, Mao03].

2.1 Abstract Algebra

In this section, we review the basic abstract algebra structures and bilinear maps.

2.1.1 Groups

Definition 2.1 (Group)**.** *A group is a non-empty set* $\mathbb{G}$ *combined with a binary operation* $\circ$ *over elements in* $\mathbb{G}$*, satisfying the following four properties.*

- **Closure**: $\forall a, b \in \mathbb{G},\ a \circ b \in \mathbb{G}$.
- **Associativity**: $\forall a, b, c \in \mathbb{G},\ (a \circ b) \circ c = a \circ (b \circ c)$.
- **Identity**: $\exists e \in \mathbb{G},\ \forall a \in \mathbb{G},\ a \circ e = e \circ a = a \in \mathbb{G}$.
- **Inverse**: $\forall a \in \mathbb{G},\ \exists a' \in \mathbb{G},\ a \circ a' = a' \circ a = e$.

If the binary operation $\circ$ is specified as addition $+$, we call such a group as additive group. If the binary operation $\circ$ is specified as multiplication $\cdot$, we call such a group as multiplicative group. For simplicity, we use $\mathbb{G}$ to denote a group with the set $\mathbb{G}$ and the binary operation $\circ$ if the binary operation $\circ$ is clear in the context.

Different notations of identity and inverse are used for groups with different operations. For additive groups, the additive identity e is denoted as 0, and inverse a' of $a \in \mathbb{G}$ is denoted as $-a$, i.e. $a - a = a + (-a) = 0$ for all $a \in \mathbb{G}$ where $\mathbb{G}$ is an additive group. For multiplicative groups, the multiplicative identity e is denoted as 1, and inverse a' of $a \in \mathbb{G}$ is denoted as a^{-1}, i.e. $a \cdot a^{-1} = 1$ for all $a \in \mathbb{G}$ where $\mathbb{G}$ is a multiplicative group. Since a group has only one operation, the equations or expressions of an additive group can be represented using a multiplicative group

by replacing the notations of the operation, the identity, and the inverse. Similarly, the equations or expressions of a multiplicative group can be represented using an additive group.

As a group contains a set, the order of a group $\mathbb{G}$ is defined by the cardinality of the internal set $\mathbb{G}$, which is denoted as $|\mathbb{G}|$. In addition, a group is a finite group if its order is finite.

Other groups with extra properties are defined as follows.

Definition 2.2 (Abelian Group)**.** *A group* $(\mathbb{G}, \circ)$ *is an abelian group if the following property is satisfied.*

- **Commutativity**: $\forall a, b \in \mathbb{G},\ a \circ b = b \circ a$.

Definition 2.3 (Cyclic Group)**.** *A group* $(\mathbb{G}, \cdot)$ *is a cyclic group if there exists an element* $g \in \mathbb{G}$ *such that* $\mathbb{G} = \{g^i \mid i \in \mathbb{Z}\}$*. Such an element* g *is called a generator of* $\mathbb{G}$*. The group* $\mathbb{G}$ *can be represented as* $\langle g \rangle$*, and called to be generated by* g*.*

Remark that the symbol $\mathbb{Z}$ refers to the set of all integers, and the Definition 2.3 also applies to groups with other operations, such as addition.

2.1.2 Rings and Fields

Definition 2.4 (Ring)**.** *A ring is an abelian group* $(\mathbb{R}, +)$ *with a binary operation* $\cdot$ *over elements in* $\mathbb{R}$*, satisfying the following four properties.*

- **Closure under multiplication**: $\forall a, b \in \mathbb{R},\ a \cdot b \in \mathbb{R}$.
- **Associativity of multiplication**: $\forall a, b, c \in \mathbb{R},\ (a \cdot b) \cdot c = a \cdot (b \cdot c)$.
- **Multiplicative identity**: $\exists e \in \mathbb{R},\ \forall a \in \mathbb{R},\ a \cdot e = e \cdot a = a \in \mathbb{R}$.
- **Distributivity**: $\forall a, b, c \in \mathbb{R},\ a \cdot (b+c) = (a \cdot b) + (a \cdot c),\ (a+b) \cdot c = (a \cdot c) + (b \cdot c)$.

Definition 2.5 (Communicative Ring)**.** *A ring* $(\mathbb{R}, +, \cdot)$ *is a communicative ring if the following property is satisfied.*

- **Commutativity of multiplication**: $\forall a, b \in \mathbb{R},\ a \cdot b = b \cdot a$.

Definition 2.6 (Field)**.** *A communicative ring* $(\mathbb{F}, +, \cdot)$ *is a field if* $(\mathbb{F} \setminus \{0\}, \cdot)$ *forms a group.*

2.1.3 Bilinear Maps

A bilinear map is a function that maps two group spaces to a third space. In this thesis, we use the same bilinear map as in [BF01] for simplicity where the first two group spaces are the same.

Definition 2.7 (Bilinear Map)**.** *Let $\mathbb{G}_1$, $\mathbb{G}_2$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}_1$. A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ has the following properties.*

- **Bilinearity**: $\forall a, b \in \{0, \dots, p-1\},\ e(g^a, g^b) = e(g, g)^{ab}$.
- **Non-Degeneracy**: $e(g, g) \neq 1$.
- **Efficiency**: It can be computed efficiently for any possible input.

2.2 Miscellaneous Notations

Let $\in_R$ denote random sampling that $x \in_R X$ means that x is uniformly and randomly chosen from a set or distribution X. Let $\lfloor x \rfloor$ denote the floor function, and $\lceil x \rceil$ denote the ceiling function of a real number x. The set $\mathbb{Z}$ represents the set of all integers. For a positive integer p, the set $\mathbb{Z}_p$ denotes the set $\{0, \dots, p-1\}$, and the notation $\mathbb{G}_p$ denotes a group of order p.

In the context of function, the symbol $\circ$ denotes the function composition that $(g \circ f)(x) = g(f(x))$. The function id denotes the identity function that $\mathsf{id}(x) = x$ and $f \circ \mathsf{id} = f$.

In the group computation, the function sca denotes the inverse operation to the scalar multiplication that $a = \mathrm{sca}_x(b) \iff b = ax$ where X is an additive group generated by x, $a \in \mathbb{Z}_{|X|}$, and $b \in X$. It is an analogue of the logarithm operation in a multiplicative group.

In the matrix computation, let $\mathbf{X}_{m \times n}$ be a matrix of size $m \times n$, and $\mathbf{I}_n$ be the identity matrix of size n. If the size is clear in the context, $\mathbf{X}$ and $\mathbf{I}$ are used for short with size omitted. Let $\top$ denote matrix transpose. In addition to the standard matrix addition and multiplication operations, we define the following notations for better representations. Let f be a function that takes an element from X with other fixed parameters as input, $\mathbf{X} \in X^{m \times n}$ be a matrix, and $\stackrel{\text{def}}{=}$ denote equal by definition. We define that

$$\mathbf{X} \stackrel{\text{def}}{=} \begin{bmatrix} X_{1,1} & \cdots & X_{1,n} \\ \vdots & \ddots & \vdots \\ X_{m,1} & \cdots & X_{m,n} \end{bmatrix}, \quad f(\mathbf{X}, \dots) \stackrel{\text{def}}{=} \begin{bmatrix} f(X_{1,1}, \dots) & \cdots & f(X_{1,n}, \dots) \\ \vdots & \ddots & \vdots \\ f(X_{m,1}, \dots) & \cdots & f(X_{m,n}, \dots) \end{bmatrix}.$$

If f is an additive homomorphism that $f(a)+f(b) = f(a+b)$, we have $f(\mathbf{A})+f(\mathbf{B}) = f(\mathbf{A}+\mathbf{B})$ where $\mathbf{A}$ and $\mathbf{B}$ are two matrices of the same size. Besides that, we can have more complex scalar multiplication as

$$\begin{aligned}
\mathbf{A}f(\mathbf{B}) &= \begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{m,1} & \cdots & A_{m,n} \end{bmatrix} \begin{bmatrix} f(B_{1,1}) & \cdots & f(B_{1,l}) \\ \vdots & \ddots & \vdots \\ f(B_{n,1}) & \cdots & f(B_{n,l}) \end{bmatrix} \\
&= \begin{bmatrix} \sum_{i=1}^{n} A_{1,i} f(B_{i,1}) & \cdots & \sum_{i=1}^{n} A_{1,i} f(B_{i,l}) \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^{n} A_{m,i} f(B_{i,1}) & \cdots & \sum_{i=1}^{n} A_{m,i} f(B_{i,l})) \end{bmatrix} \\
&= \begin{bmatrix} f(\sum_{i=1}^{n} A_{1,i} B_{i,1}) & \cdots & f(\sum_{i=1}^{n} A_{1,i} B_{i,l}) \\ \vdots & \ddots & \vdots \\ f(\sum_{i=1}^{n} A_{m,i} B_{i,1}) & \cdots & f(\sum_{i=1}^{n} A_{m,i} B_{i,l})) \end{bmatrix} = f(\mathbf{AB})
\end{aligned}$$

where $\mathbf{A}$ is an $m \times n$ matrix and $\mathbf{B}$ is an $n \times l$ matrix. Similarly, we have $f(\mathbf{A})\mathbf{B} = f(\mathbf{AB})$. In addition, the symbol $(\mathbf{A} \mid \mathbf{B})$ denotes an argumented matrix of two matrices $\mathbf{A}$ and $\mathbf{B}$ with the same row size. It can be generalised as $(\mathbf{A}_1 \mid \cdots \mid \mathbf{A}_n) \stackrel{\text{def}}{=} ((\cdots((\mathbf{A}_1 \mid \mathbf{A}_2) \mid \mathbf{A}_3) \mid \cdots) \mid \mathbf{A}_n)$.

2.3 Computational Complexity

The model of Turing machine is a general model of computation where a Turing machine consists of a finite-state control unit, a tape, and a tape head. Let $\mathcal{A}$ be an algorithm, and x be the input of $\mathcal{A}$. The computation of the Turing machine running the algorithm $\mathcal{A}$ (or simply the machine $\mathcal{A}$) on the input tape with the encodings of x is denoted by $y \leftarrow \mathcal{A}(x)$ where the result y is the output of $\mathcal{A}$. The Turing machine can run algorithms with multiple inputs by encoding all inputs on a single tape or letting M be a multi-tape Turing machine.

For the security models defined in this thesis, adversaries are modelled as oracle machines, which are Turing machines with a oracle tape and two oracle states. The computation of the machine $\mathcal{A}$ on the input x with the access to the oracle $\mathcal{O} : \{0,1\}^* \to \{0,1\}^*$ is denoted by $y \leftarrow \mathcal{A}^{\mathcal{O}}(x)$ where the machine $\mathcal{A}$ can query $\mathcal{O}(q)$ with a valid string q.

A probabilistic polynomial-time (PPT) Turing machine is a Turing machine running a randomised algorithm where the running time is bounded by a polynomial in the length of the input. An algorithm is said to be efficient or a PPT algorithm if it can be run by a PPT Turing machine.

A negligible function negl is a function that for all positive polynomial p there

exists an N such that for all $n > N$, $\mathsf{negl}(n) < 1/p(n)$. In this thesis, a function is said to be negligible if it is bounded by $O(\mathsf{negl}(\lambda))$ where λ is the security parameter in the context. Again, a function is said to be non-negligible if it is not negligible. In terms of probability, an event E has overwhelming probability if $\Pr[\neg E] = 1 - \Pr[E]$ is negligible.

2.4 Complexity Assumptions

In this section, we review the number-theoretic problems and complexity assumptions related to this thesis.

2.4.1 Standard Assumptions

The Discrete Logarithm (DL) problem is one of the most fundamental number-theoretic problems in cryptography.

Definition 2.8 (Discrete Logarithm problem)**.** *Let $\mathbb{G}$ be a cyclic multiplicative group of prime order p where $|p| = \lambda$ and λ is the security parameter. Let $x \in_R \mathbb{Z}_p$, and $g \in_R \mathbb{G}$. Given g and g^x, there is an algorithm $\mathcal{A}$ that computes x with advantage:*

$$\mathsf{Adv}^{\mathrm{DL}}_{\mathcal{A}} = \Pr[x \leftarrow \mathcal{A}(g, g^x)].$$

By analysing the DL problem in the generic group model (see Section 2.5.5), Shoup [Sho97] gives a theorem that $\mathsf{Adv}^{\mathrm{DL}}_{\mathcal{A}}$ is bounded by $O(q^2/p)$ where q is the number of group operations. Therefore, the DL problem is computationally hard for any PPT algorithm $\mathcal{A}$ since $\mathsf{Adv}^{\mathrm{DL}}_{\mathcal{A}}$ is negligible.

Based on the DL problem, Diffie and Hellman [DH76] introduced a new problem, which is known as the Computational Diffie-Hellman (CDH) problem, and apply it to their public-key cryptosystem.

Definition 2.9 (Computational Diffie-Hellman problem)**.** *Let $\mathbb{G}$ be a cyclic multiplicative group of prime order p where $|p| = \lambda$ and λ is the security parameter. Let $a, b \in_R \mathbb{Z}_p$, and $g \in_R \mathbb{G}$. Given g, g^a and g^b, there is an algorithm $\mathcal{A}$ that computes g^{ab} with advantage:*

$$\mathsf{Adv}^{\mathrm{CDH}}_{\mathcal{A}} = \Pr[g^{ab} \leftarrow \mathcal{A}(g, g^a, g^b)].$$

There is a decisional variant of the CDH problem named Decisional Diffie-Hellman (DDH) problem, and many encryption schemes are based on this problem. For instance, the ElGamal encryption scheme [ElG85] and Cramer-Shoup encryption schemes [CS98, CS02] are well-known encryption schemes based on the DDH problem.

Definition 2.10 (Decisional Diffie-Hellman problem)**.** *Let $\mathbb{G}$ be a cyclic multiplicative group of prime order p where $|p| = \lambda$ and λ is the security parameter. Let $a, b \in_R \mathbb{Z}_p$, and $g, Z \in_R \mathbb{G}$. Given two probability distributions $\mathcal{D}_{\mathrm{DDH}} = \{(g, g^a, g^b, g^{ab})\}$ and $\mathcal{D}_R = \{(g, g^a, g^b, Z)\}$, there is an algorithm $\mathcal{A}$ that distinguishes $\mathcal{D}_{\mathrm{DDH}}$ and $\mathcal{D}_R$ with advantage:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{DDH}} = \left|\Pr[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_{\mathrm{DDH}})] - \Pr[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_R \setminus \mathcal{D}_{\mathrm{DDH}})]\right|.$$

Similarly to the DL problem, Shoup [Sho97] points out that the advantage $\mathsf{Adv}^{\mathrm{CDH}}$ and $\mathsf{Adv}^{\mathrm{DDH}}$ are both bounded by $O(q^2/p)$, which is negligible. Hence, the CDH problem and the DDH problem are computationally hard for any PPT algorithm $\mathcal{A}$.

Due to the extra group operation of bilinear maps (Definition 2.7), the hardness of the DDH problem does not hold. Instead, Boneh and Franklin [BF01] introduced a natural variant of the DDH problem suitable for groups with bilinear maps named Decisional Bilinear Diffie-Hellman (DBDH) problem, and proposed an Identity-Based Encryption (IBE) scheme based on this problem.

Definition 2.11 (Decisional Bilinear Diffie-Hellman problem)**.** *Let $\mathbb{G}_1, \mathbb{G}_2$ be two cyclic multiplicative groups of prime order p where $|p| = \lambda$ and λ is the security parameter, and $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a bilinear map. Let $a, b, c \in_R \mathbb{Z}_p$, $g \in_R \mathbb{G}_1$, and $Z \in_R \mathbb{G}_2$. Given two probability distributions $\mathcal{D}_{\mathrm{DBDH}} = \{(g, g^a, g^b, g^c, e(g,g)^{abc})\}$ and $\mathcal{D}_R = \{(g, g^a, g^b, g^c, Z)\}$, there is an algorithm $\mathcal{A}$ that distinguishes $\mathcal{D}_{\mathrm{DBDH}}$ and $\mathcal{D}_R$ with advantage:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{DBDH}} = \left|\Pr[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_{\mathrm{DBDH}})] - \Pr[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_R \setminus \mathcal{D}_{\mathrm{DBDH}})]\right|.$$

The advantage can be represented alternatively as

$$D_0 \in_R \mathcal{D}_{\mathrm{DBDH}}, \quad D_1 \in_R \mathcal{D}_R, \quad b \in_R \{0,1\},$$
$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{DBDH}} = \left|\Pr[b = b' \leftarrow \mathcal{A}(D_b)] - \frac{1}{2}\right|.$$

By adapting the generic group model to generic bilinear group model, Boneh et al. [BBG05] shows the lower bound of the DBDH problem in terms of computational complexity, indicating the DBDH problem is computationally hard for any PPT algorithm $\mathcal{A}$.

There is another variant of the DDH problem regarding to bilinear maps named Decision Linear (DLIN) problem introduced by Boneh et al. [BBS04].

Definition 2.12 (Decision Linear problem)**.** *Let $\mathbb{G}_1, \mathbb{G}_2$ be two cyclic multiplicative groups of prime order p where $|p| = \lambda$ and λ is the security parameter, and $e : \mathbb{G}_1 \times$*

$\mathbb{G}_1 \to \mathbb{G}_2$ *be a bilinear map. Let* $a, b \in_R \mathbb{Z}_p$*, and* $g, h, f, Z \in_R \mathbb{G}_1$*. Given two probability distributions* $\mathcal{D}_{\text{DBDH}} = \{(g, g^a, h, h^b, f, f^{a+b})\}$ *and* $\mathcal{D}_R = \{(g, g^a, h, h^b, f, Z)\}$*, there is an algorithm* $\mathcal{A}$ *that distinguishes* $\mathcal{D}_{\text{DLIN}}$ *and* $\mathcal{D}_R$ *with advantage:*

$$\mathsf{Adv}_{\mathcal{A}}^{\text{DLIN}} = \left|\Pr[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_{\text{DLIN}})] - \Pr[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_R \setminus \mathcal{D}_{\text{DLIN}})]\right|.$$

Boneh et al. [BBS04] claim that the DLIN problem is computationally hard in the generic bilinear group model for any PPT algorithm $\mathcal{A}$. Furthermore, the DLIN problem is also captured by the Decisional Bilinear (P, f)-Diffie-Hellman ((P, f)-DBDH) problem in Section 5.2, which gives the same result as [BBS04].

2.4.2 Subset Membership Problem

Subset Membership Problem (SMP) is a problem class introduced by Cramer and Shoup [CS02]. Many standard decisional problems can be classified to SMP, including DDH, DLIN, and Decisional Composite Residuosity (DCR) problems.

Definition 2.13 (Subset Membership Problem)**.** *Let* $L \subset X, W$ *be three non-empty sets, and* $R = \{(x, w) \mid x \in L\} \subset X \times W$ *be a binary relation where* w *is a witness of a word* x *in the language* L*. Let* $\Lambda = (X, L, W, R)$ *be a problem instance,* $x \in_R L$*, and* $x' \in_R X \setminus L$*. Given two probability distributions* $\mathcal{D}_L = \{(\Lambda, x)\}$ *and* $\mathcal{D}_{X\setminus L} = \{(\Lambda, x')\}$*, there is an algorithm* $\mathcal{A}$ *that distinguishes* $\mathcal{D}_L$ *and* $\mathcal{D}_{X\setminus L}$ *with advantage:*

$$\mathsf{Adv}_{\mathcal{A}}^{\text{SMP}} = \left|\Pr[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_L)] - \Pr[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_{X\setminus L})]\right|.$$

An SMP is computationally hard if and only if the advantage $\mathsf{Adv}_{\mathcal{A}}^{\text{SMP}}$ is negligible for any PPT algorithm $\mathcal{A}$. As our schemes in this thesis are based on DDH and DCR problems, the details of how they are captured by SMP are shown as follows.

Definition 2.14 (Decisional Diffie-Hellman problem as Subset Membership Problem)**.** *Let* $\mathbb{G}$ *be a cyclic multiplicative group of prime order* p *where* $|p| = \lambda$ *and* λ *is the security parameter. Let* $r \in_R \mathbb{Z}_p$*, and* $g_1, g_2, Z_1, Z_2 \in_R \mathbb{G}$*. Given two probability distributions* $\mathcal{D}_{\text{DDH}} = \{(g_1, g_2, g_1^r, g_2^r)\}$ *and* $\mathcal{D}_R = \{(g_1, g_2, Z_1, Z_2)\}$*, there is an algorithm* $\mathcal{A}$ *that distinguishes* $\mathcal{D}_{\text{DDH}}$ *and* $\mathcal{D}_R$ *with advantage:*

$$\mathsf{Adv}_{\mathcal{A}}^{\text{DDH}} = \left|\Pr[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_{\text{DDH}})] - \Pr[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_R \setminus \mathcal{D}_{\text{DDH}})]\right|.$$

The DCR problem is introduced by Paillier [Pai99] for the Paillier cryptosystem. Cramer and Shoup [CS02] adapt the original DCR problem to the SMP class as follows.

Definition 2.15 (Decisional Composite Residuosity problem)**.** *Let p, q be two safe primes such that $p = 2p' + 1$ and $q = 2q' + 1$ where p', q' are two primes of length λ bites and λ is the security parameter. Let $N = pq$, and $N' = p'q'$. We have that $\mathbb{Z}^*_{N^2} = \mathbb{G}_N\mathbb{G}_{N'}\mathbb{G}_2\mathbb{G}_T$ where $\mathbb{G}_T$ is a group generated by $-1 \pmod{N^2}$. Let $P = \mathbb{G}_{N'}\mathbb{G}_2\mathbb{G}_T \subset \mathbb{Z}^*_{N^2}$, $x \in_R P$, and $x' \in_R \mathbb{Z}^*_{N^2} \setminus P$. Given two probability distributions $\mathcal{D}_P = \{(N, x)\}$ and $\mathcal{D}_{\mathbb{Z}^*_{N^2}\setminus P} = \{(N, x')\}$, there is an algorithm $\mathcal{A}$ that distinguishes $\mathcal{D}_P$ and $\mathcal{D}_{\mathbb{Z}^*_{N^2}\setminus P}$ with advantage:*

$$\mathsf{Adv}^{\mathrm{DCR}}_{\mathcal{A}} = \left|\Pr[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_P)] - \Pr[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_{\mathbb{Z}^*_{N^2}\setminus P})]\right|.$$

2.5 Cryptographic Tools

In this section, we review the cryptographic tools used in the construction and the security proof of our schemes.

2.5.1 Secret Sharing Scheme

Shamir's secret sharing scheme [Sha79] divides a secret s into n pieces $s_1, \ldots, s_n$ using a $k-1$ degree polynomial and distributes to n users. If and only if k users or more come together, they can recover s by polynomial interpolation. Knowing $k-1$ pieces of s does not reveal any information about s. This scheme is also called (k, n) *Threshold Secret Sharing Scheme*. Details are shown as follows.

Let $\mathbb{F}_q$ be a finite field of order q where $q > n$. Each user U_i is associated with a public unique number $u_i \in \mathbb{F}_q$. To share a secret $s \in \mathbb{F}_q$ among a user set $S = \{U_1, \ldots, U_n\}$, a random $k-1$ degree polynomial is picked as $p(x) = s + \sum_{j=1}^{k-1} a_j x^j$ where $a_j \in_R \mathbb{F}_q$. Each user in the user set S gets a share $s_i = p(u_i)$. When k users come together and form a user set $A \subset S$, we can recover $p(x) = \sum_{U_i \in A} \Delta^A_i s_i$ where $\Delta^A_i = \prod_{U_\ell \in A \wedge i \neq \ell} \frac{x - u_\ell}{u_i - u_\ell}$. Then we can recover $s = p(0)$. Obviously, we can recover any point by

$$s_j = p(u_j) = \sum_{U_i \in A} \Delta^A_{ij} s_i \quad \text{where } \Delta^A_{ij} = \prod_{U_\ell \in A \wedge i \neq \ell} \frac{u_j - u_\ell}{u_i - u_\ell}.$$

By defining $u_0 = 0$, we have $s = \sum_{U_i \in A} \Delta^A_{i0} s_i$.

2.5.2 Cryptographic Hash Functions

Hash functions are efficient functions mapping a bit string of arbitrary length to a fixed-length bit string [Gol01, Mao03]. Formally, we consider *universal hash families* [CW79, WC81, Dam88, Gol01, CS02] with the definition as follows.

Definition 2.16 (Universal Hash Family)**.** *A universal hash family consists of the following three polynomial time algorithms:*

- $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$*: The randomised system setup algorithm takes a security parameter* 1^λ *as input. It specifies the hash domain* X*, the hash codomain* Π*, and the key space* K*. Then it publishes the system parameter* $\mathsf{param} = (X, K, \Pi)$*.*
- $k \leftarrow \mathsf{KeyGen}(\mathsf{param})$*: The randomised hash key generation algorithm takes system parameter* param *as input, and outputs a randomly chosen hash key* $k \in_R K$*.*
- $\pi \leftarrow \mathsf{Hash}(k, x)$*: The deterministic evaluation algorithm takes a hash key* k *and a value* $x \in X$*, and outputs a hash value* $\pi \in \Pi$*.*

In [Dam88, Gol01], the algorithms Setup and KeyGen are defined to be one single algorithm that outputs a random hash function H_k where k is a hash key and $H_k(\,\cdot\,) \stackrel{\text{def}}{=} \mathsf{Hash}(k, \cdot\,)$. There is a special type of hash function named Collision Resistant Hash Function (CRHF). The adapted definition from [Dam88, Gol01] is shown as follows.

Definition 2.17 (Collision Resistant Hash Function)**.** *A universal hash family is collision resistant if the advantage* $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{CRHF}}$ *of any PPT algorithm* $\mathcal{A}$ *is negligible.*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{CRHF}} = \Pr\left[\mathsf{Hash}(k, x_0) = \mathsf{Hash}(k, x_1) \;\middle|\; \begin{array}{c} k \leftarrow \mathsf{KeyGen}(\mathsf{Setup}(1^\lambda)), \\ (x_0, x_1) \leftarrow \mathcal{A}(k) \wedge x_0 \neq x_1 \end{array}\right].$$

A hash function derived from a collision resistant universal hash family is called a collision resistant hash function.

2.5.3 Hash Proof Systems

By extending the universal hash family, Cramer and Shoup [CS02] introduced the notion of universal projective hash family and further extend it to Hash Proof System (HPS).

Definition 2.18 (Hash Proof System)**.** *A Hash Proof System (HPS) associated with subset membership problems consists of the following five polynomial time algorithms:*

- $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$*: The randomised system setup algorithm takes a security parameter* 1^λ *as input, and specifies an SMP instance* $\Lambda = (X, L, W, R)$ *where the hash domain is* X*. The algorithm further specifies a secret hash key space* K*, a public hash key space* S*, and a hash codomain* Π*. After that, the algorithm packs all above descriptions and publishes a system-wide parameter* $\mathsf{param} = (X, L, W, R, K, S, \Pi)$*.*

- $\mathsf{SK} \leftarrow \mathsf{SKGen}(\mathsf{param})$*: The randomised secret hash key generation algorithm takes a system parameter* param*, and generates a random secret hash key* $\mathsf{SK} \in_R K$*.*
- $\mathsf{PK} \leftarrow \mathsf{PKGen}(\mathsf{SK})$*: The deterministic public hash key generation algorithm takes a secret hash key* $\mathsf{SK} \in K$ *as input, and maps it to a public hash key* $\mathsf{PK} \in S$*.*
- $\pi \leftarrow \mathsf{Hash}(\mathsf{SK}, x)$*: The deterministic private evaluation algorithm takes a secret hash key* $\mathsf{SK} \in K$ *and a word* $x \in X$ *as inputs, and outputs a hash value* $\pi \in \Pi$ *of* x*.*
- $\pi \leftarrow \mathsf{PHash}(\mathsf{PK}, x, w)$*: The deterministic public evaluation algorithm takes a public hash key* $\mathsf{PK} \in S$ *and a word* x *in the language* L *along with a corresponding witness* $w \in W$ *such that* $(x, w) \in R$*, and output a hash value* $\pi \in \Pi$ *of* x*.*

A HPS is required to be correct that the private evaluation algorithm Hash and the public evaluation algorithm PHash are equivalent when $x \in L$.

Definition 2.19 (Correctness)**.** *A HPS is correct if*

$$\forall \mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda), \quad \forall \mathsf{SK} \leftarrow \mathsf{SKGen}(\mathsf{param}), \quad \mathsf{PK} \leftarrow \mathsf{PKGen}(\mathsf{SK}),$$
$$\forall (x, w) \in R, \quad \mathsf{Hash}(\mathsf{SK}, x) = \mathsf{PHash}(\mathsf{PK}, x, w).$$

Furthermore, some useful security properties of a HPS are required.

Definition 2.20 (Universal)**.** *A HPS is universal if the following probability is negligible for all* $\mathsf{PK} \in S$*,* $x \in X \setminus L$ *and* $\pi \in \Pi$*.*

$$\mathsf{Adv}^{\text{Universal}} = \Pr[\mathsf{Hash}(\mathsf{SK}, x) = \pi \mid \mathsf{PKGen}(\mathsf{SK}) = \mathsf{PK}].$$

Definition 2.21 (Universal$_2$)**.** *A HPS is universal*$_2$ *if the following probability is negligible for all* $\mathsf{PK} \in S$*,* $x^* \in X$*,* $x \in X \setminus (L \cup \{x^*\})$ *and* $\pi^*, \pi \in \Pi$*.*

$$\mathsf{Adv}^{\text{Universal}_2} = \Pr[\mathsf{Hash}(\mathsf{SK}, x) = \pi \mid \mathsf{Hash}(\mathsf{SK}, x^*) = \pi^* \wedge \mathsf{PKGen}(\mathsf{SK}) = \mathsf{PK}].$$

If a HPS is universal$_2$ and $|X| > 1$, it is also universal.

Definition 2.22 (Smoothness)**.** *A HPS is smooth if the advantage* $\mathsf{Adv}_{\mathcal{A}}^{\text{Smooth}}$ *of an adversary* $\mathcal{A}$ *distinguishing two random variables* (x, PK, π) *and* (x, PK, π') *is negligible where* $\mathsf{SK} \in_R K$*,* $\mathsf{PK} = \mathsf{PKGen}(\mathsf{SK})$*,* $x \in_R X \setminus L$*,* $\pi = \mathsf{Hash}(\mathsf{SK}, x)$*, and* $\pi' \in_R \Pi$*.*

$$\mathsf{Adv}_{\mathcal{A}}^{\text{Smooth}} = |\Pr[1 \leftarrow \mathcal{A}(x, \mathsf{PK}, \pi)] - \Pr[1 \leftarrow \mathcal{A}(x, \mathsf{PK}, \pi')]|$$

2.5.4 Random Oracle Model

Bellare and Rogaway [BR93] introduced the notion of random oracle model, which is an idealised model of cryptography hash functions.

In this model, the hash function is treated as a black box that the computation of the hash value is only available via oracle queries. More precisely, the simulator $\mathcal{S}$ in a security game maintains a list $\mathcal{H}$ of pairs (w, h) for a hash function H. To obtain the hash value of x, the adversary $\mathcal{A}$ has to query $\mathcal{S}$ for the value x. Upon receiving x, the simulator $\mathcal{S}$ searches the list $\mathcal{H}$. If there is an entry (w, h) that $x = w$, the simulator $\mathcal{S}$ answers $\mathcal{A}$ with the hash value h. Otherwise, the simulator $\mathcal{S}$ randomly and uniformly chooses a value h as the hash value $H(x)$ of x, and answers $\mathcal{A}$. At the same time, the simulator $\mathcal{S}$ appends the newly created pair (x, h) to the list $\mathcal{H}$ for later use. Since the random oracle is controlled by the simulator $\mathcal{S}$, the oracle can be programmed to answer $\mathcal{A}$ with any value as long as the distribution is uniformed. This makes random oracle model extremely powerful that the simulator can embed the hard problem into the random oracle and thus the security proof can be simplified. As a result, the schemes designed in the random oracle model are much more efficient than those designed in the standard model.

However, negative results have been shown by Canetti et al. [CGH98] that there exist cryptosystems which are proven secure in the random oracle model but insecure in the standard model.

2.5.5 Generic Group Model

Similar to the random oracle model for the hash functions, Shoup [Sha79] introduced the notion of generic group model, which is an idealised model of group.

In this model, the group operation is treated as a black box that the computation of the group operation is only available via oracle queries. Like the random oracle model, the simulator $\mathcal{S}$ maintains a list $\mathcal{L} = \{(p, \varepsilon)\}$ of pairs of a polynomial and a uniformly random encoding string. At the beginning of the simulation, the simulator $\mathcal{S}$ sends all encoding strings of the initial list $\mathcal{L}$ without the corresponding polynomial. To operate on the group, the adversary $\mathcal{A}$ is required to query the group operation oracle with two encoding strings $\varepsilon_1, \varepsilon_2$ and the type of the group operation. The restriction for $\mathcal{A}$ is that $\varepsilon_1, \varepsilon_2$ must be on the list $\mathcal{L}$. Once the simulator $\mathcal{S}$ receives the query, it computes $p_3 = p_1 + p_2$ or $p_3 = p_1 - p_2$, depending on whether the operation is addition/multiplication or subtraction/division. If p_3 is on the list $\mathcal{L}$, the simulator $\mathcal{S}$ answers with the corresponding encoding string. Otherwise, it randomly and uniformly chooses a new encoding string ε_3, appends the pair (p_3, ε_3) to the list $\mathcal{L}$, and answers $\mathcal{A}$ with ε_3.

Differing from the random oracle model, the group operation oracle is not pro-

grammable. It is used for probability analysis and computing the lower bound of the computational complexity of a number-theoretic problem. As mentioned in Section 2.4.1, Shoup [Sha79] proved that the DL problem, the CDH problem, and the DDH problem are computationally hard in the generic group model.

The generic group model is only suitable for single groups. For groups with bilinear maps, Boneh et al. [BBG05] adapt the generic group model to the generic bilinear group model. In this model, the simulator $\mathcal{S}$ maintains different lists for different groups. Besides the group operation oracles for different groups, the simulator $\mathcal{S}$ also provides a bilinear pairing oracle for bilinear maps between groups. As mentioned in Section 2.4.1, the generic bilinear group model has been applied to prove the hardness of the DBDH problem and the DLIN problem.

Similar to random oracle model, the generic group model is not a perfect model that a lower bound analysed in the generic groups does not imply a lower bound in any specific group.

Chapter 3

Exploration of Practical Functional Encryption

In the chapter, we review the development and the formal definition of Functional Encryption (FE), and explore the practical functionalities for FE.

3.1 Introduction

3.1.1 A Brief History

In 1976, Diffie and Hellman [DH76] opened a new era in cryptography. Instead of sharing a same secret key with both the sender Alice and the receiver Bob in applying cryptography, Bob has a pair of secret key and public key. In Public Key Encryption (PKE), Alice uses Bob's public key to encrypt messages as ciphertexts. Then Bob obtains the original messages (plaintexts) by decrypting the ciphertexts with his secret key.

However, public keys consume a large amount of storage in a large system. Moreover, public keys need to be certified by a trusted third party in order to prevent impersonation. In order to simplify public keys, Shamir [Sha85] gave an idea of using the users' identities (e.g. name or email address, etc.) as public keys where the corresponding secret keys are generated by a Key Generation Centre (KGC). Unfortunately, Shamir only proposed an Identity-Based Signature scheme with his idea in 1985 without any Identity-Based Encryption (IBE) implementation. Later in 2001, Boneh and Franklin [BF01] first proposed a fully functional IBE scheme with proven semantical security against an adaptive chosen ciphertext attack (IND-ID-CCA), assuming the hardness of the Computational Diffie-Hellman (CDH) problem in elliptic curve groups with bilinear pairing in random oracle model. With such an IBE scheme, Alice can use Bob's identity to encrypt messages under the public key of the KGC. Later, Bob asks the KGC to obtain his secret key for his identity. Finally, Bob decrypts the ciphertexts with his secret key. Unlike PKE, Alice can encrypt messages before Bob getting his secret key.

Alone with the development of IBE, Sahai and Waters [SW05] introduced Fuzzy

Identity-Based Encryption scheme also known as Attribute-Based Encryption (ABE) scheme. In ABE, attributes and policies are presented as identities. If the policy is associated with the user's keys and the attributes are associated with the ciphertext, this type of ABE is named Key-Policy Attribute-Based Encryption (KP-ABE) [SW05]. For instance, Alice encrypts messages with attributes "UOW" and "Academy". If Bob has a secret key for a policy "UOW or Student", he can decrypt the ciphertext as the attributes in the ciphertext match the policy associated with Bob's secret key. Another type of ABE is Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [BSW07] where the attributes are associated with the user's keys and the policy is associated with the ciphertext. Different from KP-ABE, Alice encrypts messages with a policy "UOW or Student" in CP-ABE and Bob can decrypt with the key associated with attributes "UOW". The policy is original implemented in a threshold manner [SW05] which has restricted representation. For better representation of the policy, Goyal et al. [GPSW06] proposed a KP-ABE scheme with fine-grained access control where the policy is implemented using Monotone Span Programs, allowing "AND" and "OR" gates in the access trees.

All previous ABE schemes [BSW07, GPSW06, SW05] are proven secure with selective security. Leveraging the power of Dual Pairing Vector Space (DPVS) [OT11], Okamoto and Takashima [OT12] proposed a fully (adaptively) secure unbounded ABE scheme under Decision Linear (DLIN) assumption in the standard model, including KP-ABE and CP-ABE. They also extended the span program in the construction from monotone to non-monotone in order to allow "NOT" gates.

In addition, one drawback of ABE scheme is that the encryption algorithm comprises complex computations and it is impractical to compute for the device with low computation power. Hohenberger and Waters [HW14] proposed an online/offline ABE that the devices can encrypt messages quickly in the online phase with the intermediate values prepared in the offline phase.

Another drawback of ABE is the weakened identity that secret key leakage cannot be traced to individual users since the relation between the users and the attributes is many-to-many. Liu et al. [LCW13a] proposed a White-Box Traceable CP-ABE scheme that the corresponding user can be traced with a secret key given. Later, Liu et al. [LCW13b] enhanced the traceability by proposing a Blackbox Traceable CP-ABE scheme. With the blackbox traceability, the corresponding user can be traced with a decryption oracle associated an unknown secret key.

From the concept of ABE, a generalised notion named Functional Encryption (FE) is introduced [BSW11, Wat13] where this term is first mentioned by Lewko et al. [LOS+10] in 2010. Later in 2011, Boneh et al. [BSW11] defined Functional Encryption that the intended user can only learn the output of a function of the message and the function key from a ciphertext. In other words, the revelation of

the information encrypted in the ciphertext can be controlled by the data owner issuing secret key for different functions while the information retrieval for conventional encryptions is all or nothing. For instance, Alice encrypts a message m under Bob's public key to produce the target ciphertext c. Bob generates a decryption key sk for a function key k using his own secret key and gives sk to Charlie. Receiving the ciphertext c and the decryption key sk, Charlie is able to learn the output of the pre-defined function with inputs m and k. Boneh et al. [BSW11] also showed that Functional Encryption models Predicate Encryption (PE), which further models IBE and ABE. Furthermore, Goldwasser et al. [GGG+14] extends Functional Encryption to Multi-Input Functional Encryption (MIFE) that arbitrary functions f are accepted and f can take multiple message inputs when decrypting multiple ciphertexts.

Under the class of FE, researchers proposed efficient schemes for new functionalities. Waters [Wat12] proposed a Deterministic Finite Automata (DFA) based encryption as an improvement of KP-ABE where the number of the attributes in the ciphertext is not fixed. For practical functionalities, Abdalla et al. [ABDCP15] proposed an efficient FE scheme for inner product evaluation. However, the security of their scheme is very weak and cannot be used in practice. Later, the succeeding works [ALS16, ABCP16, ZMY17] improve the scheme to IND-CPA security and further IND-CCA security. In the direction of MIFE, Abdalla et al. [ARW16] showed that a secure MIFE scheme of inner products can be constructed from a Functional Encryption for Inner Products (FE-IP) scheme trivially in the public-key setting.

Another interesting cryptographic primitive modelled by FE is Public-key Encryption with Keyword Search (PEKS) introduced by Boneh et al. [BDCOP04]. Precisely, a PEKS scheme is equivalent to a FE scheme for equality test functionality.

3.1.2 Chapter Organisation

The rest of this chapter is organised as follows. In Section 3.2, we review the formal definition of FE and its security model. Then we discuss FE schemes for practical functionalities in Section 3.2, including PE, equality test, inequality test, and inner product evaluation. Finally, the summary is addressed in Section 3.4.

3.2 Formal Definitions

The definition and the security model of functional encryption by Boneh et al. [BSW11] are reviewed as follows.

3.2.1 Syntax

Definition 3.1 (Functional Encryption)**.** *Let $f : K \times X \to Y$ be a universal function for a function class $\mathcal{F} = \{f_k \mid k \in K\}$ indexed by a function key space K, mapping the message space X to the revelation space Y. A Functional Encryption (FE) for a function class $\mathcal{F}$ consists of the following four polynomial time algorithms:*

- $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$*: The randomised system setup algorithm takes a security parameter 1^λ as input, and generates system-wide parameters and a random pair of a master secret key* MSK *and a public key* PK*.*
- $\mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, k)$*: The (probably) randomised secret key generation algorithm takes a master secret key* MSK *and a function key $k \in K$ as inputs, and computes a secret key* SK *for the function f_k.*
- $C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, x)$*: The randomised encryption algorithm takes a public key* PK *and a message $x \in X$ as inputs, and generates a ciphertext C of the message x.*
- $y \leftarrow \mathsf{Decrypt}(\mathsf{SK}, C)$*: The (probably) deterministic decryption algorithm takes a secret key* SK *for a function f_k and a ciphertext C of a message x as inputs. The algorithm reveals $y = f_k(x) \in Y$ from the ciphertext C, and outputs it.*

Since f is modelled as a universal function, we may use $f(k, x)$ to denote $f_k(x)$ in this thesis.

3.2.2 Security Model

Indistinguishability-based security is considered in this thesis. Precisely, we consider the Indistinguishability under Chosen Plaintext Attacks (IND-CPA) formulated by Boneh et al. [BSW11]. In the IND-CPA game (Fig. 3.1), an adaptive IND-CPA adversary $\mathcal{A}$ tries to distinguish a target ciphertext from two messages x_0 and x_1 chosen by $\mathcal{A}$ as follows.

Setup phase The challenger $\mathcal{S}$ runs the system setup algorithm $\mathsf{Setup}(1^\lambda)$ to generate a key pair $(\mathsf{MSK}, \mathsf{PK})$, and passes the public key PK to the adversary $\mathcal{A}$.

Pre-challenge phase The adversary $\mathcal{A}$ can adaptively query the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}$ with a function key $k \in K$ to obtain a secret key SK for the function f_k. The restriction is that the adversary $\mathcal{A}$ can only query the function key k such that $f_k(x_0) = f_k(x_1)$ where x_0 and x_1 are the messages chosen by $\mathcal{A}$ in the challenge phase. Otherwise, winning the game is trivial since $\mathsf{Decrypt}(\mathsf{SK}, C) \neq f_k(x_{1-b})$.

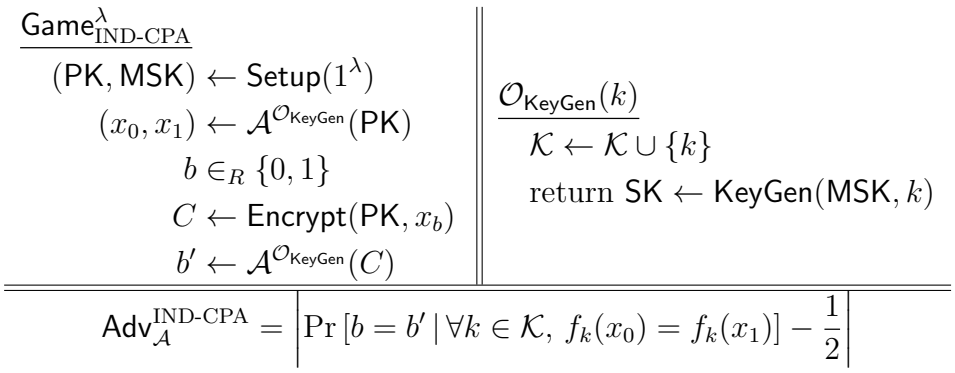

Figure 3.1: IND-CPA Game

Challenge phase At some point, the adversary $\mathcal{A}$ outputs two messages x_0 and x_1. The challenger $\mathcal{S}$ randomly picks $b \in_R \{0, 1\}$, and generates a target ciphertext $C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, x_b)$. After that, the challenger $\mathcal{S}$ passes the target ciphertext C to the adversary $\mathcal{A}$.

Post-challenge phase The adversary $\mathcal{A}$ can further query the oracle $\mathcal{O}_{\mathsf{KeyGen}}$ as in the pre-challenge phase with the same restriction.

Guessing phase Eventually, the adversary $\mathcal{A}$ outputs an educated guess b'. If $b = b'$, the adversary $\mathcal{A}$ wins the game.

The advantage of the adversary $\mathcal{A}$ winning the IND-CPA game (Fig. 3.1) is

$$\mathsf{Adv}^{\text{IND-CPA}}_{\mathcal{A}} = \left| \Pr\left[b = b' \mid \forall k \in \mathcal{K},\ f_k(x_0) = f_k(x_1)\right] - \frac{1}{2} \right|$$

Definition 3.2 (IND-CPA Security)**.** *An FE scheme is Indistinguishable under Chosen Plaintext Attacks (IND-CPA) if the advantage* $\mathsf{Adv}^{\text{IND-CPA}}_{\mathcal{A}}$ *for all adversary* $\mathcal{A}$ *winning the IND-CPA game (Fig. 3.1) in the polynomial time is a negligible function.*

3.3 Discussions on Practical Functionalities

In the series of works after [BSW11], FE schemes for general circuits [GGH+13, GKP+13] are proposed. However, those schemes are not sufficiently efficient to be applied in practical scenarios. On the other hand, real-world users are interested in the specific functionalities in practical instead of general circuits. Therefore, we look into efficient FE constructions for practical functionalities, which are useful in the real world.

In this section, we review the definition of PE and its subclasses. Then we dig into the functionalities of equality and inequality tests for both PE and FE. For

inequality tests, we propose a minor improvement on the efficiency for PE with public index. Besides that, we review the functionality of inner product evaluation and related functionalities.

3.3.1 Predicate Encryption

Definition 3.3 (Predicate Encryption)**.** *Let $P : K \times I \to \{0,1\}$ be a predicate function where I is the index space and M is the message payload space. Predicate Encryption (PE) [KSW08] is a subclass of FE with the functionality*

$$f(k \in K, (i,m) \in X = (I \times M)) = \begin{cases} m & \text{if } P(k,i) = 1, \\ \perp & \text{otherwise.} \end{cases}$$

Since the PE is a subclass of FE, all the syntax and security models are compatible. For less confusion, the **attribute hiding** security for PE is equivalent to the IND-CPA security defined for FE.

As mentioned in [BSW11], there is a variant of PE that the index $i \in I$ is always public (i.e. without attribute hiding).

Definition 3.4 (Predicate Encryption with Public Index)**.** *Let $P : K \times I \to \{0,1\}$ be a predicate function where I is the index space and M is the message payload space. Predicate Encryption with public index is a subclass of FE with the functionality*

$$f(k \in K, (i,m) \in X = (I \times M)) = \begin{cases} (i,m) & \text{if } P(k,i) = 1, \\ i & \text{otherwise.} \end{cases}$$

PE schemes intend to apply access control for decryption ability of the payload message. Based on the precise predicates, various cryptographic primitives can be classified in to the class of FE, notably IBE [BF01, Gen06], ABE [SW05, GPSW06, BSW07, LOS+10, OT12], Hidden Vector Encryption (HVE) [BW07], and Inner Product Encryption (IPE) [KSW08, LOS+10, OT15].

Definition 3.5 (Identity-Based Encryption)**.** *Identity-Based Encryption (IBE) is an instance of PE with public index with the predicate*

$$P(k \in K, i \in I) = \begin{cases} 1 & \text{if } k = i, \\ 0 & \text{otherwise.} \end{cases}$$

Anonymous Identity-Based Encryption (AnonIBE) is an instance of PE with the same predicate.

There are two types of ABE named KP-ABE [SW05] and CP-ABE [BSW07].

Definition 3.6 (Key-Policy Attribute-Based Encryption)**.** *Let K be the set of all policies, and I be the power set of the set of all attributes. Key-Policy Attribute-Based Encryption (KP-ABE) is an instance of PE with public index with the predicate*

$$P(k \in K, i \in I) = \begin{cases} 1 & \text{if } k \text{ accepts } i, \\ 0 & \text{otherwise.} \end{cases}$$

Definition 3.7 (Ciphertext-Policy Attribute-Based Encryption)**.** *Let K be the power set of the set of all attributes, and I be the set of all policies. Ciphertext-Policy Attribute-Based Encryption (CP-ABE) is an instance of PE with public index with the predicate*

$$P(k \in K, i \in I) = \begin{cases} 1 & \text{if } k \text{ is accepted by } i, \\ 0 & \text{otherwise.} \end{cases}$$

Like ABE, there are two types of IPE named Zero Inner Product Encryption (ZIPE) [KSW08] and Non-zero Inner Product Encryption (NIPE) [OT15]. Conventionally, the name IPE is referred as ZIPE. Let $K = I = \mathbb{V}$ be the same vector space.

Definition 3.8 (Zero Inner Product Encryption)**.** *Zero Inner Product Encryption (ZIPE) is an instance of PE with the predicate*

$$P(\vec{k} \in \mathbb{V}, \vec{i} \in \mathbb{V}) = \begin{cases} 1 & \text{if } \vec{k} \cdot \vec{i} = 0, \\ 0 & \text{otherwise.} \end{cases}$$

Definition 3.9 (Non-zero Inner Product Encryption)**.** *Non-zero Inner Product Encryption (NIPE) is an instance of PE with the predicate*

$$P(\vec{k} \in \mathbb{V}, \vec{i} \in \mathbb{V}) = \begin{cases} 1 & \text{if } \vec{k} \cdot \vec{i} \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

Let $K = (\mathbb{G} \cup \{*\})^n$, $I = \mathbb{G}^n$ be two vector spaces of the same dimension n where $\mathbb{G}$ is a group and $*$ is a wildcard symbol.

Definition 3.10 (Hidden Vector Encryption)**.** *Hidden Vector Encryption (HVE) is an instance of PE with the predicate*

$$P(\vec{k} \in K, \vec{i} \in I) = \begin{cases} 1 & \text{if } \forall j \in \{1, \ldots, n\},\ k_j = * \vee k_j = i_j \\ 0 & \text{otherwise.} \end{cases}$$

3.3.2 Equality Test

Definition 3.11 (Functional Encryption for Equality Tests)**.** *Functional Encryption for Equality Tests (FE-ET) is a subclass of FE with the functionality*

$$f(k \in K, x \in X) = \begin{cases} 1 & \text{if } k = x, \\ 0 & \text{otherwise.} \end{cases}$$

By observing Definitions 3.5 and 3.11, it is found that AnonIBE implies FE-ET. Precisely, they are the same if $|M| = 1$ in AnonIBE. In addition, the notion of FE-ET is equivalent to PEKS introduced by Boneh et al. [BDCOP04] both for the syntax and the security model, i.e. IND-CPA security for FE-ET is equivalent to Indistinguishability under Adaptive Chosen Keyword Attacks (IND-CKA) security for PEKS. Abdalla et al. [ABC$^+$08] concluded the same result that AnonIBE is equivalent to PEKS and thus is equvalent to FE-ET.

Hence, FE-ET is a good start point to research searchable encryptions, which are useful in cloud computing applications. For further research related to searchable encryptions, we direct the reader to Part II of this thesis.

3.3.3 Inequality Test

Generally speaking, inequality is a relation that two elements are not equal. Precisely, it is more meaningful if the elements are in an ordered set so that they can be compared. Let S be an ordered set with a total order $\leq$ and $a, b \in S$. The equality test $a = b$ is defined as $a \leq b \wedge b \leq a$, and the strict order $a < b$ is defined as $\neg(b \leq a)$. Besides, we also define $a > b$ if $b < a$, and $a \geq b$ if $b \leq a$.

Definition 3.12 (Functional Encryption for Inequality Tests)**.** *Let the infix* $\diamond \in \{<, \leq, \geq, >\}$ *denote a general comparison operator, and* $K = X = S$*. Functional Encryption for Inequality Tests (FE-IT) is a subclass of FE with the functionality*

$$f_\diamond(k \in S, x \in S) = \begin{cases} 1 & \text{if } x \diamond k, \\ 0 & \text{otherwise.} \end{cases}$$

Besides the FE for pure inequality test functionality, we are also interested in the PE with the inequality test predicate.

Definition 3.13 (Predicate Encryption for Inequality Tests)**.** *Let the infix* $\diamond \in \{<, \leq, \geq, >\}$ *denote a general comparison operator, and* $K = I = S$*. Predicate*

Encryption for Inequality Tests (PE-IT) is an instance of PE with the predicate

$$P_\diamond(k \in S, i \in S) = \begin{cases} 1 & \text{if } i \diamond k, \\ 0 & \text{otherwise.} \end{cases}$$

Similar to the relation of FE-ET and AnonIBE, by observing Definitions 3.12 and 3.13, it is clear that PE-IT implies FE-IT. Therefore, a PE-IT scheme can be built in order to obtain FE-IT.

Before the discussion of detailed implementations of FE-IT, there is a fundamental problem of "how to compare two integer". For example, how to evaluate $a < b$. Boneh and Waters [BW07] suggest a number-line like method that they prepare a set $B = \{1, 2, \ldots, b-2, b-1\}$ for b and then test whether $a \in B$ or not. This method is workable but inefficient with both the time and space complexity of $O(n)$ since they actually encode the integers in unary and test in unary. As a better solution, Bethencourt et al. [BSW07] encode the integers in the binary form and compare them bit by bit, reducing both the time and space complexity to $O(\log n)$. However, they still need to do the actual computation of the evaluation that each bit evaluation requires reconstructions of secret-sharing schemes, which makes the multiplicative factor of the complexity large. In addition, their computation is based on boolean expression evaluation. If we want to obtain a PE-IT scheme via ZIPE, the expression must be converted to conjunctive normal form (CNF) or disjunctive normal form (DNF) so that the expression evaluation can be converted into an inner product evaluation However, this operation may lead to an exponential explosion, making the complexity worse than $O(n)$.

Trying to further increase the efficiency, we leverage the idea from *automata theory* [Sip96]. By observing the syntax of the PE-IT with $P_\diamond(k, i)$, we find out that the operation $\diamond$ and the value k associated with secret key will never be changed once the secret key is generated. For the ciphertext, we also find out that the value i associated will never be changed once generated. Therefore, to maximise the efficiency, it is nature to compile $P_\diamond(k, \cdot)$ into a finite automaton $M_{\diamond,k}$ that takes i as the input. However, we still need to do the actual computation by running the machine $M_{\diamond,k}$, and it requires that both the machine and input are public and not obfuscated that the alphabet symbols $\{0, 1\}$ are necessary to carry out a state transfer. Hence, we compile the inputs and alter the original automaton $M_{\diamond,k}$ so that only equality tests are required instead of state transfers for computing. More precisely, we compile the value k into a set A of prefixes of all binary strings i such that $P_\diamond(k, i) = 1$, and compile i into a set B of prefixes of its binary form. To compute $P_\diamond(k, i)$, all we need is to compute $A \cap B \stackrel{?}{=} \emptyset$.

However, the problem occurs when we implement the computation of $A \cap B \stackrel{?}{=}$

$$
\begin{array}{l||l}
\underline{\mathsf{Game}^{\lambda}_{\mathrm{MR}}} & \\
\quad (\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^{\lambda}) & \underline{\mathcal{O}_{\mathsf{KeyGen}}(k)} \\
\quad\quad (x_0, x_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}}}(\mathsf{PK}) & \quad \mathcal{K} \leftarrow \mathcal{K} \cup \{k\} \\
\quad\quad\quad b \in_R \{0,1\} & \quad \text{return } \mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, k) \\
\quad\quad\quad C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, x_b) & \\
\quad\quad\quad b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}}}(C) &
\end{array}
$$

$$
\mathsf{Adv}^{\mathrm{MR}}_{\mathcal{A}} = \left| \Pr\left[b = b' \mid \forall k \in \mathcal{K},\ f_k(x_0) = f_k(x_1) = 0 \right] - \frac{1}{2} \right|
$$

Figure 3.2: MR Game

$\emptyset$ using a ZIPE scheme [KSW08]. Since [KSW08] accepts boolean expressions in polynomials only, the above intersection test operation has to be converted to a polynomial with the degree of $d = O((\log n)^2)$ and $t = O(\log n)$ variables. The resulted complexity is $d^t = O((\log n)^{2 \log n})$, which is worse than $O(n)$. Therefore, this approach is a negative result.

As an independent work, Shi et al. [SBC+07] defined an interval tree to represent the integer universe. Similarly to our approach from automata theory, the authors use sets of labels to represent integers and ranges, and conduct the evaluation by computing whether the intersection is an empty set or not. They built their FE-IT scheme on top of a AnonIBE scheme with decryption cost of $O((\log n)^2)$ (or $O(\log n)$ with optimisations). However, their scheme cannot achieve the standard IND-CPA security for FE. Instead, the authors introduced a weaker security model named Match-Revealing (MR) as illustrated in Fig. 3.2, and referred the standard IND-CPA security as Match-Concealing (MC). Note that the key query for the function f_k with the condition $f_k(x_0) = f_k(x_1) = 1$ is allowed in the IND-CPA game but disallowed in the MR game. The authors proved the security of their scheme in the selective version of the MR security model. Therefore, it is a security-efficiency trade-off for FE-IT.

Nevertheless, our approach from automata theory is still able to increase the efficiency of PE-IT related schemes with public index. In CP-ABE [BSW07], the comparison operation of integers is done bit by bit, which requires $O(\log n)$ reconstructions of secret-sharing schemes. With our approach, we reduce the number of reconstructions from $O(\log n)$ to $O(1)$. More precisely, computing $A \cap B \stackrel{?}{=} \emptyset \iff |A \cap B| \stackrel{?}{\geq} 1$ requires only one secret-sharing reconstruction with an OR gate (i.e. a (k, n) secret-sharing scheme with threshold $k = 1$), since the sets A and B are public.

In the rest of this subsection, we present our detailed encoding technique of compiling integers and functions $P_{\diamond}(k, \cdot)$ into sets as follows, starting from basic binary encodings.

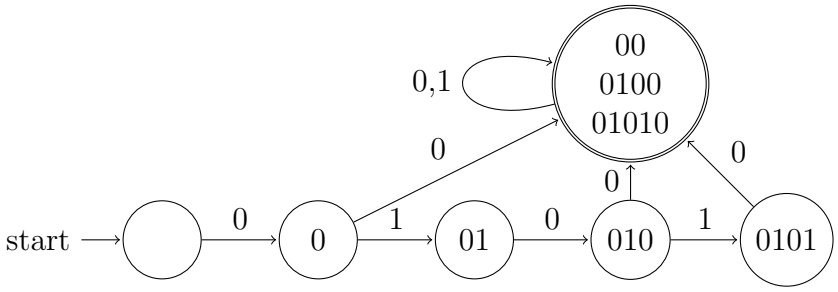

Figure 3.3: Deterministic Finite Automata $M_{<,22}$ for $P_<(22, \cdot)$

Let $S = \mathbb{Z}_n$ be a finite integer set where $|S| = n$, and $\ell = \lceil \log_2 n \rceil$. Therefore, every element $a \in S$ can be rewritten in the binary form with the length of ℓ. For instance, $22 \in \mathbb{Z}_{42}$ in the binary form is 010110 of length $\ell = 6$.

The encoding (Algorithm 1) for the machine input x is to generate all prefixes of its binary form in different sizes. For instance, $\mathsf{ENCODE}(22) \rightarrow T_{22} = \{0, 01, 010, 0101, 01011, 010110\}$. The size of the result set of $\mathsf{ENCODE}(x)$ is ℓ.

Algorithm 1 Encoding for x

Require: x in the binary form of the length ℓ.

```
1: procedure ENCODE(x)
2:     T ← ∅                          ▷ T is initialized as an empty set
3:     t ← ""                         ▷ t is initialized as an empty binary string
4:     for i = 1 to ℓ do
5:         t ← t || x_i               ▷ Append the ith bit of x to t and save to t
6:         T ← T ∪ {t}                ▷ Add t to T
7:     end for
8:     return T
9: end procedure
```

Our scheme supports four operations $\{<, \leq, \geq, >\}$. We propose the corresponding encodings as follows. The encoding (Algorithm 2) for the machine $P_<(k, \cdot)$ is to generate all paths to the accepting state of the original machine $M_{<,k}$. In other words, it generates all shortest-size prefixes of all binary strings x such that the prefixes represent $x < k$. For instance, $\mathsf{ENCLESS}(22) \rightarrow T_{22}^{<} = \{00, 0100, 01010\}$, which are the paths to the accepting states of the original machine $M_{<,22}$ (Fig. 3.3). The encoding (Algorithm 4) for the machine $P_>(k, \cdot)$ is to generate all shortest-size prefixes of all binary strings x such that the prefixes represent $x > k$. For instance, $\mathsf{ENCGREATER}(22) \rightarrow T_{22}^{>} = \{1, 011, 010111\}$. The size of the result sets of $\mathsf{ENCLESS}(k)$ or $\mathsf{ENCGREATER}(k)$ are at most ℓ. The encoding (Algorithm 3) for the machine $P_{\leq}(k, \cdot)$ and the encoding (Algorithm 5) for the machine $P_{\geq}(k, \cdot)$ add k itself to the results of Algorithms 2 and 4 for non-strict inequalities. For instance, $\mathsf{ENCLEQ}(22) \rightarrow T_{22}^{\leq} = \{00, 0100, 01010, 010110\}$ and $\mathsf{ENCGEQ}(22) \rightarrow T_{22}^{\geq} = \{1, 011, 010110, 010111\}$.

Algorithm 2 Encoding for $P_<(k, \cdot)$

Require: k in the binary form of the length ℓ.

procedure ENCLESS(k)
  $T \leftarrow \emptyset$ ▷ T is initialised as an empty set
  $t \leftarrow$ “” ▷ t is initialised as an empty binary string
  for $i = 1$ to ℓ **do**
    if $k_i = 1$ **then** ▷ k_i means the ith bit of k
      $T \leftarrow T \cup \{t||0\}$ ▷ Append 0 to t and add the result to T
    end if
    $t \leftarrow t \,||\, k_i$ ▷ Append k_i to t and save to t
  end for
  return T
end procedure

Algorithm 3 Encoding for $P_\leq(k, \cdot)$

Require: k in the binary form of the length ℓ.

procedure ENCLEQ(k)
  $T \leftarrow$ ENCLESS(k) ▷ Call Algorithm 2
  $T \leftarrow T \cup \{k\}$ ▷ Add k to T
  return T
end procedure

Algorithm 4 Encoding for $P_>(k, \cdot)$

Require: k in the binary form of the length ℓ.

procedure ENCGREATER(k)
  $T \leftarrow \emptyset$ ▷ T is initialised as an empty set
  $t \leftarrow$ “” ▷ t is initialised as an empty binary string
  for $i = 1$ to ℓ **do**
    if $k_i = 0$ **then** ▷ k_i means the ith bit of k
      $T \leftarrow T \cup \{t||1\}$ ▷ Append 1 to t and add the result to T
    end if
    $t \leftarrow t \,||\, k_i$ ▷ Append k_i to t and save to t
  end for
  return T
end procedure

Algorithm 5 Encoding for $P_\geq(k, \cdot)$

Require: k in the binary form of the length ℓ.

procedure ENCGEQ(k)
  $T \leftarrow$ ENCGREATER(k) ▷ Call Algorithm 4
  $T \leftarrow T \cup \{k\}$ ▷ Add k to T
  return T
end procedure

In addition, we prove that our encoding scheme is correct. Let $\mathsf{CODEX} = \{(<, \mathsf{ENCLESS}), (\leq, \mathsf{ENCLEQ}), (\geq, \mathsf{ENCGEQ}), (>, \mathsf{ENCGREATER})\}$.

Theorem 3.1. *Let* $a, b \in \mathbb{Z}_n$, $(\diamond, \mathsf{ENC}_\diamond) \in \mathsf{CODEX}$, $A = \mathsf{ENCODE}(a)$, *and* $B_\diamond = \mathsf{ENC}_\diamond(b)$. *We have* $a \diamond b \iff A \cap B_\diamond \neq \emptyset$.

Proof. We prove this theorem in two steps.

- $a \diamond b \implies A \cap B_\diamond \neq \emptyset$: From Algorithm 2, the set $B_\diamond$ contains all shortest-size prefixes of all binary strings x such that the prefixes represent $x \diamond k$. Thus if $a \diamond b$, the $B_\diamond$ contains a prefix of a of some size. Since A contains its prefixes of all sizes (Algorithm 1), we have $A \cap B_\diamond \neq \emptyset$.
- $A \cap B_\diamond \neq \emptyset \implies a \diamond b$: Since the set $B_\diamond$ contains all shortest-size prefixes of all binary strings x such that the prefixes represent $x \diamond k$ and one of the prefix of a exists in $B_\diamond$, we immediately have $a \diamond b$.

□

Some readers may try to take intersection or union of encoded sets to achieve conjunctive or disjunctive expressions. However, that may not work. For example, $T_3^> = \mathsf{ENCGREATER}(3) = \{1, 01, 001, 0001\}$ and $T_5^< = \mathsf{ENCLESS}(5) = \{0000, 000100\}$, we have $T_3^> \cap T_5^< = \emptyset$ instead of $T_4^= = \{000100\}$. Moreover, as an extension of our encoding technique, we still can take unions of encoded sets if the intended sets have no intersection.

3.3.4 Inner Product Evaluation

Definition 3.14 (Functional Encryption for Inner Products)**.** *Let* $K = X = \mathbb{V}$ *be the same vector space. Functional Encryption for Inner Products (FE-IP) [ABDCP15] is a subclass of FE with the functionality*

$$f(\vec{k} \in \mathbb{V}, \vec{x} \in \mathbb{V}) = \langle \vec{k}, \vec{x} \rangle = \vec{k} \cdot \vec{x}.$$

Differing from IPE, which invokes inner product evaluation for the predicate, FE-IP computes the inner product as the decryption result. The detailed definition of FE-IP is available in Section 6.2. As mentioned by Abdalla et al. [ABDCP15], the main practical application for FE-IP is to calculate the weighted mean or the weighted sum in descriptive statistics.

In this thesis, we propose a FE-IP with enhanced security in Chapter 6 and generalise the functionality of inner product evaluation to matrix product evaluation in Chapter 7.

Definition 3.15 (Functional Encryption for Matrix Products)**.** *Let* $K = X = \mathbb{M}$ *be the same matrix space, and* $\mathbf{K} \in \mathbb{M}$, $\mathbf{X} \in \mathbb{M}$ *be two matrices such that the number of columns in* $\mathbf{K}$ *is equal to the number of rows in* $\mathbf{K}$*. Functional Encryption for Matrix Products is a subclass of FE with the functionality*

$$f(\mathbf{K} \in \mathbb{M}, \mathbf{X} \in \mathbb{M}) = \mathbf{KX}.$$

3.4 Chapter Summary

In chapter, we reviewed the history of Functional Encryption in the literature. Besides, we also reviewed the definition of FE and its IND-CPA security model. For the main part, we pointed out the practical functionalities as the subclasses of FE, including Predicate Encryption, Functional Encryption for Equality Tests, Functional Encryption for Inequality Tests, and Functional Encryption for Inner Products.

As a minor contribution, we introduced an encoding technique that can be used in the inequality tests, enhancing the efficiency of CP-ABE from $O(\log n)$ to $O(1)$ secret-sharing reconstructions in decryption.

Part II

Keyword Search Techniques in Cloud Computing

Chapter 4

Threshold Broadcast Encryption with Keyword Search

Many users store their data in an untrusted cloud for the purpose of convenient data access and sharing. For convenience, keyword search can be performed by the cloud remotely with a single query from the user. However, the cloud cannot directly search the data if it is encrypted for security purposes. In order to avoid local searches by downloading all data from the cloud, causing a huge amount of network bandwidth and computation power consumption of local devices, keyword search on encrypted data has been proposed. Searchable encryption schemes make outsourcing the search operation with privacy possible. In this chapter, a special variant of searchable encryption with threshold access is studied. Unlike some previous proposals which have a fixed group and a fixed threshold value, we define a new notion named Threshold Broadcast Encryption with Keyword Search (TBEKS) for dynamic groups and flexible threshold values. We formalise the security of a TBEKS scheme via a new security model named Indistinguishability in the threshold setting under Chosen Keyword Attacks (IND-T-CKA). We also propose the first practical TBEKS scheme with provable security in our IND-T-CKA security model, assuming the hardness of the Decisional Bilinear Diffie-Hellman (DBDH) problem. Parts of this work appeared in [ZMY16].

4.1 Introduction

Cloud computing [MG11] provides flexible computing resources, including data storage, to end users. Users are able to upload their data to the cloud for later access by themselves or by other users (i.e., data sharing) via the Internet. In other words, on-demand data access is available via the Internet where users can search and then download what they need. To prevent a huge amount of network bandwidth consumption, the search operations are usually done by the cloud instead of letting users download all the data and search locally.

Meanwhile, to ensure the privacy of the users, some sensitive data should be protected against the cloud server while the keyword search functionality is main-

tained. Specifically, the data to be searched and the keyword used in the search operation should be inaccessible by any non-authorised parties, including the cloud. With such a demand, various searchable encryption schemes [ABC+08, BSNS06, BDCOP04, HL07, SWP00] have been proposed to enable secure searching over encrypted data. In a public key searchable encryption scheme, Bob encrypts both the data and the keywords under Alice's public key and uploads the ciphertexts to the cloud. As both the data and the keywords are protected, it is hard for the cloud to gain any information about the data. To perform search operations, Alice generates a trapdoor for a keyword [ABC+08, BSNS06, BDCOP04] or multiple keywords [HL07] and transfers it to the cloud via a secure communication channel. Upon receiving Alice's trapdoor of the keyword, the cloud searches the whole database and returns the search results back to Alice. Finally, Alice downloads the ciphertexts from the cloud based on the search results, and decrypts them to get the original data.

In the normal searchable encryption schemes, the accessibility to the data and its search operation is authorised to a user [ABC+08, BDCOP04] or a set of users [SYL+14, ZXA14] where any *single* user in the authorised user set can perform the search and the decryption operations. However, a single identity may not be trustful in some scenarios. For instance, a research team of a company is developing a new product and needs to access the company database. The head of the research department does not trust any single member of the research team to access the database, since an individual member may leak the secrets of the company for monetary purposes. To reduce the risk of a single point failure, a threshold searchable encryption scheme is more suitable where the accessibility to the data is decentralised from a single member to n members of the team where searching the database and decrypting a ciphertext both require at least t members to work together. To be more precise, in order to perform a search operation successfully, the cloud needs to obtain for a keyword at least t trapdoors from the n authorised users. If such a threshold searchable encryption also supports dynamic groups and flexible threshold values, the company can specify different classifications for different data by changing the authorised user set and the threshold value t. This chapter aims to provide a practical solution for this problem.

4.1.1 Related Work

Boneh et al. [BDCOP04] introduced the searchable encryption, namely Public-key Encryption with Keyword Search (PEKS), and defined the security model that the adversary cannot identify the keyword from the ciphertext without a trapdoor. Xu et al. [XJWW13] argued that PEKS is insecure under Keyword Guessing Attack

(KGA) since the remote server can always create a ciphertext of a keyword and test it with the target trapdoor. If the keyword space is in polynomial size, the adversary can get the keyword from the target trapdoor in polynomial time. Xu et al. [XJWW13] also proposed a method to enhance the security under KGA by encrypting and searching for the fuzzy keyword instead of the exact keyword.

Besides, Boneh et al. [BDCOP04] showed that PEKS implies Identity-Based Encryption (IBE) but not vice versa. Nevertheless, Abdalla et al. [ABC+08] proposed a PEKS scheme generically constructed using Anonymous Identity-Based Encryption (AnonIBE). They also proposed Identity-Based Searchable Encryption (IBKS) from a 2-leveled Anonymous Hierarchical Identity-Based Encryption (AnonHIBE). Similarly, searchable broadcasting encryption, namely Broadcast Encryption with Keyword Search (BEKS), can be constructed using 2-leveled Anonymous Hierarchical Identity-Coupling Broadcast Encryption (AnonHICBE) [AFI06].

Searchable encryption can be divided into the single user setting (e.g. PEKS) and the multi-user setting. Broadcast Encryption with Keyword Search [AFI06] and Attribute-Based Encryption with Keyword Search (ABKS) [SYL+14, ZXA14] are in the multi-user setting. In BEKS, the keyword is encrypted for a set of users. If a user is in the target set, the user can generate the trapdoor for testing the ciphertext. In ABKS, the keyword is encrypted under a policy or with attributes. Only the user who has a match of the policy and the attributes can generate the trapdoor for testing the ciphertext.

However, in both BEKS and ABKS, the individual target user has the full ability to generate the trapdoor. Wang et al. [WWP08] decentralised the ability of trapdoor generation to multi-user in a threshold manner, which requires at least k of n user to generate the trapdoor. Siad [Sia12] gave a formal definition of Threshold Public-key Encryption with Keyword Search (TPEKS), and generically constructed a TPEKS scheme with threshold (n, t)-IBE but no concrete scheme is provided. In Wang et al.'s scheme [WWP08], a trusted centralised manager is required to generate the private keys for all users. To enhance the security, Siad's scheme [Sia12] leverages a distributed protocol in private keys generation instead of a trusted third party.

We find that both schemes [Sia12, WWP08] are limited to a fixed number of users and a fixed threshold value at the key generation stage. It makes adding or removing a user impossible, and changing the threshold value for individual ciphertext impossible. To encrypt a keyword for different set of users or with different threshold value, we have to generate the private keys for all the users in the target set. If it is an (n, t)-TPEKS scheme where t is the threshold value such that $0 < t \leq n$ and n is the maximum number of users, the users have to store $O(n \cdot 2^n)$ private-public key pairs for the possible ciphertexts, although they may share the same global public parameters.

4.1.2 Our Contribution

In this chapter, we introduce a new notion named Threshold Broadcast Encryption with Keyword Search (TBEKS). We provide a formal definition of TBEKS and a formal security model, named Indistinguishability in the threshold setting under Chosen Keyword Attacks (IND-T-CKA), to capture its security. Moreover, we construct a practical TBEKS scheme and prove that it is IND-T-CKA secure under the Decisional Bilinear Diffie-Hellman (DBDH) assumption in random oracle model.

In our TBEKS definition and scheme, users are ad hoc, i.e., they can generate their own private-public key pair individually. The data owner selects a target set of users and threshold value t to encrypt a keyword, and then uploads the full ciphertext to the remote server. To search the files containing a certain keyword, at least t users of the target user set need to generate their trapdoor shares for that keyword, and transfer those trapdoor shares to the remote server, in order to enable the remote server to perform the search operation. Our scheme does not fix the user group and the threshold value at the system setup, and only one private-public key pair is required for each user. Thus we solve Wang et al.'s open problem for dynamic group [WWP08].

4.1.3 Chapter Organisation

The rest of this chapter is organised as follows. We define TBEKS and its security model in Section 4.2. Then we propose our TBEKS scheme in Section 4.3 and prove it is secure under the security model defined in Section 4.2.2. Finally, a summary is addressed in Section 4.5.

4.2 Formal Definitions

Generally speaking, a Threshold Broadcast Encryption with Keyword Search (TBEKS)[1] scheme is used along with a Threshold Broadcast Encryption (TBE) scheme [DHMR07], where the former encrypts the keywords and the latter encrypts the message[2]. Independent private-public key pairs are suggested for the combination of the above mentioned system. In TBEKS, there are three roles involved, including the **data owner** who encrypts the message and the keywords, the **server** who stores the ciphertexts and performs the requested search, and the **user** who has the access to the decryption of the message and generates search queries. TBEKS works as follows. The **data owner** chooses a set of **users** and a threshold value t, and encrypts the message under TBE and the keyword under TBEKS. Then the **data owner**

[1] We choose the name TBEKS in order to separate it from TPEKS.

[2] In a storage system, messages are actually files.

combines the ciphertexts and uploads them to the **server**. To perform a search operation, at least t **users** generate their individual trapdoors for the same target keyword W and upload the trapdoors to the **server** via a secure communication channel. After that, the **server** searches the whole database of ciphertexts with the given trapdoors and returns the result message indices back. Upon receiving the indices, the **users** retrieve the corresponding ciphertexts and decrypt them with at least t **users** working together. Note that only the trapdoors are required to be transferred via a secure communication channel.

4.2.1 Syntax

Formally, we present the definition of TBEKS as follows.

Definition 4.1 (Threshold Broadcast Encryption with Keyword Search)**.** *A Threshold Broadcast Encryption with Keyword Search (TBEKS) scheme, involving the data users, the servers and the users U_i, consists of the following five possibly probabilistic polynomial time algorithms:*

- $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$*: The randomised system setup algorithm takes a security parameter 1^λ, and outputs a set of parameters used in the system widely. This algorithm can be run by anyone whereas all users are required to agree on the same parameters.*
- $(\mathsf{PK}_i, \mathsf{SK}_i) \leftarrow \mathsf{KeyGen}(\mathsf{param})$*: The randomised user key generation algorithm takes a system parameter* param*, and outputs a pair of secret key SK_i and public key PK_i of a user U_i. This algorithm is run by the users individually.*
- $C \leftarrow \mathsf{TBEKS}(\{\mathsf{PK}_1, \ldots, \mathsf{PK}_n\}, t, W)$*: The randomised keyword encryption algorithm takes a set of public keys $\{\mathsf{PK}_1, \ldots, \mathsf{PK}_n\}$ of n target users, a threshold value t and a keyword W, and outputs a ciphertext C of the keyword W. This algorithm is run by the data owner.*
- $T \leftarrow \mathsf{Trapdoor}(\mathsf{SK}_i, W)$*: The possibly randomised trapdoor generation algorithm takes the secret key SK_i of a user U_i and a keyword W, and outputs a user trapdoor T of the keyword W. This algorithm is run by the users individually.*
- $1/0 \leftarrow \mathsf{Test}(\{T_1, \ldots, T_t\}, C)$*: The deterministic test algorithm takes t trapdoors $T_i \leftarrow \mathsf{Trapdoor}(\mathsf{SK}_i, W)$ and a keyword ciphertext $C \leftarrow \mathsf{TBEKS}(\{\mathsf{PK}'_1, \ldots, \mathsf{PK}'_n\}, t', W')$, and outputs*

$$\begin{cases} 1 & \text{if } W = W' \wedge t \geq t' \wedge \{\mathsf{PK}_1, \ldots, \mathsf{PK}_n\} \subset \{\mathsf{PK}'_1, \ldots, \mathsf{PK}'_n\}, \\ 0 & \text{otherwise.} \end{cases}$$

where $(\mathsf{PK}_i, \mathsf{SK}_i) \leftarrow \mathsf{KeyGen}(params)$. *This algorithm is run by the servers and the results will be sent back to each user involved.*

In addition, we require the scheme to be correct.

Definition 4.2 (Correctness)**.** *A TBEKS scheme is correct if the following statement is always true:*

$$\forall \mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda), \quad \forall (\mathsf{SK}, \mathsf{PK}) \leftarrow \mathsf{KeyGen}(\mathsf{param}),$$
$$\forall n, t \in \mathbb{Z}^+ \wedge t \leq n, \quad \forall W \in \{0,1\}^*, \quad \forall C \leftarrow \mathsf{TBEKS}(\{\mathsf{PK}_1, \ldots, \mathsf{PK}_n\}, t, W),$$
$$\forall S \subset \{1, \ldots, n\} \wedge t \leq |S| \leq n, \quad \mathsf{Test}(\{T \mid T \leftarrow \mathsf{Trapdoor}(\mathsf{SK}_i, W) \wedge i \in S\}, C) = 1.$$

4.2.2 Security Model

In Definition 4.1, we implicitly allow the server to combine the trapdoors for a keyword freely without any interaction with related users. For instance, the data owner creates $C_1 \leftarrow \mathsf{TBEKS}(\{\mathsf{PK}_1, \mathsf{PK}_2\}, 2, W)$. To search for C_1, the users U_1, U_2 generate $T_i \leftarrow \mathsf{Trapdoor}(\mathsf{SK}_i, W)$ for $i = 1, 2$. Later, the data owner creates $C_2 \leftarrow \mathsf{TBEKS}(\{\mathsf{PK}_2, \mathsf{PK}_3\}, 2, W)$ for the same keyword W. Similarly, to search for C_2, the users U_2, U_3 generate $T_i' \leftarrow \mathsf{Trapdoor}(\mathsf{SK}_i, W)$ for $i = 2, 3$. If $\mathsf{Trapdoor}$ is a deterministic algorithm, the server can easily link T_1, T_2 and T_3' together that they are created for the same keyword since $T_2 = T_2'$. As a result, if the data owner creates $C_3 \leftarrow \mathsf{TBEKS}(\{\mathsf{PK}_1, \mathsf{PK}_2, \mathsf{PK}_3\}, 3, W)$, the server can search C_3 by $\mathsf{Test}(\{T_1, T_2, T_3'\}, C_3) = 1$. However, the server gains no information, especially the keyword encrypted in the trapdoors, other than the test result. Instead, this provides a feature that the server can cache the uploaded trapdoors from the users.

Because of this feature, we consider that the server is *honest but curious*. Importantly, we do not allow the server to collude with any users. Otherwise, the user U_1 and the server can learn the keyword in the ciphertext. For example, the server gets T_1, T_2 and T_3 from the users to test C_3. Then the server can use T_2 and T_3 to test C_2, and return the result to the user U_1. Now, the user U_1 knows the keyword of a ciphertext while U_1 is not in the target user set.

We also do not consider the Keyword Guessing Attack (KGA) [XJWW13], since the server can always create a ciphertext $C = \mathsf{TBEKS}(\{\mathsf{PK}\}, 1, W)$ for all the keywords W with the user's trapdoor T. Commonly, if the keyword space is polynomial sized, the server can get the corresponding keyword W of C in polynomial time. However, this kind of attack can be prevented by Xu et al.'s method [XJWW13]. Boneh et al.'s scheme [BDCOP04] also does not consider this attack.

In TBEKS schemes, many users are involved. We consider that all users are registered before creating the ciphertexts, as the adversary may be able to register the

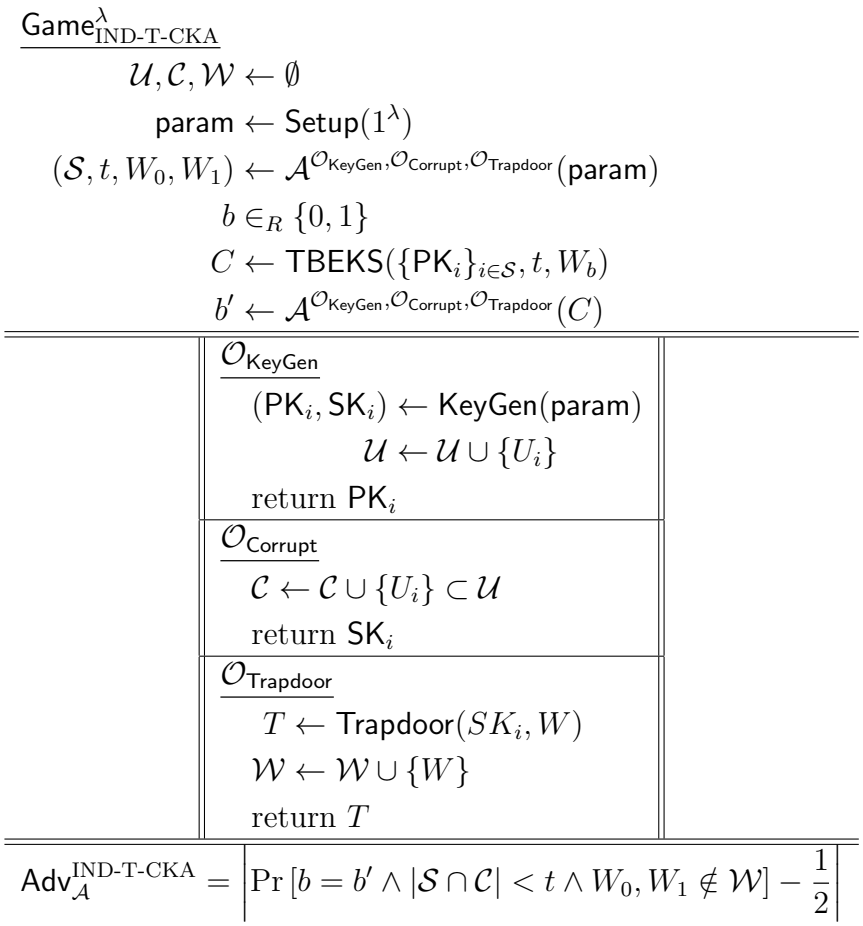

Figure 4.1: IND-T-CKA Game

private-public key pair of the target user. Now we define the Indistinguishability in the threshold setting under Chosen Keyword Attacks (IND-T-CKA) game (Fig. 4.1) where an active adversary $\mathcal{A}$ tries to distinguish two encryptions of keywords W_0 and W_1 with the security parameter λ:

1. The challenger runs the $\mathsf{Setup}(1^{\lambda})$ algorithm to generate a set of system-wide parameters and passes them to the adversary $\mathcal{A}$.

2. The adversary can adaptively ask the challenger to register a user and obtain the public key of that user by querying the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}$. At the same time, the challenger records the requested user U_i in the user list $\mathcal{U}$.

3. The adversary can adaptively ask the challenger to obtain the secret key of a registered user $U_i \in \mathcal{U}$ by querying the collusion oracle $\mathcal{O}_{\mathsf{Corrupt}}$. At the same time, the challenger records the requested user U_i in the collusion list $\mathcal{C}$.

4. The adversary can adaptively ask the challenger to obtain the user U_i's trapdoor of a keyword W by querying the trapdoor generation oracle $\mathcal{O}_{\mathsf{Trapdoor}}$. At the same time, the challenger records the requested keyword W in the keyword list $\mathcal{W}$. For the corrupted users $U_i \in \mathcal{C}$, the adversary can compute the

trapdoor by itself using the users U_i's secret key SK_i. Hence, the keyword list $\mathcal{W}$ only contains the requested keywords of uncorrupted users.

5. At some point, the adversary $\mathcal{A}$ outputs a set $\mathcal{S}$ of users, a threshold value t and two keywords W_0 and W_1 to be challenged. The adversary is restricted that W_0 and W_1 are not in the list $\mathcal{W}$ as they are not queried to $\mathcal{O}_{\mathsf{Trapdoor}}$. The adversary is also restricted that it cannot corrupt t users or more in the user set $\mathcal{S}$.

6. The challenger randomly selects b to be 0 or 1, and gives a ciphertext $C = \mathsf{TBEKS}(\{\mathsf{PK}_i\}_{i\in\mathcal{S}}, t, W_b)$ to the adversary $\mathcal{A}$.

7. The adversary can continue to query all three oracles $\mathcal{O}_{\mathsf{KeyGen}}$, $\mathcal{O}_{\mathsf{Corrupt}}$, $\mathcal{O}_{\mathsf{Trapdoor}}$ with the same restrictions.

8. Eventually, the adversary $\mathcal{A}$ outputs a bit b'. If $b = b'$, the adversary wins the game.

We define the advantage of winning the IND-T-CKA game (Fig. 4.1) as

$$\mathsf{Adv}_{\mathcal{A}}^{\text{IND-T-CKA}} = \left| \Pr\left[b = b' \wedge |\mathcal{S} \cap \mathcal{C}| < t \wedge W_0, W_1 \notin \mathcal{W}\right] - \frac{1}{2} \right|.$$

Definition 4.3 (IND-T-CKA Security)**.** *A TBEKS scheme is indistinguishable in the threshold setting against chosen keyword attack (IND-T-CKA) if* $\mathsf{Adv}_{\mathcal{A}}^{\text{IND-T-CKA}}$ *is a negligible function for all adversary* $\mathcal{A}$ *winning the the IND-T-CKA game (Fig. 4.1) in polynomial time.*

4.3 The Construction

We build our TBEKS scheme based on Daza et al.'s TBE scheme [DHMR07] using the idea similar to Boneh et al.'s PEKS scheme [BDCOP04]. The main idea of the construction is to use the secret keys of users as the shares of a shared secret in an $(n, 2n - t)$ threshold secret sharing scheme. The shared secret works as the secret key of a dummy user in [BDCOP04]. Since all computations are done with points on the elliptic curve and due to the hardness of Discrete Logarithm (DL) problem, the secret shares are computationally secure.

Our TBEKS scheme works as follows.

– $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$: Given a security parameter 1^λ, this algorithm generates a prime number of q bits and multiplicative groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of order q where there is a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. This algorithm also picks a random generator g of $\mathbb{G}_1$. After that, the algorithm picks two hash functions

$H : \{0,1\}^* \to \mathbb{G}_1$ and $H' : \mathbb{G}_1 \to \mathbb{Z}_q$. Note that the hash function H is used to hash a keyword W into an element in $\mathbb{G}_1$, and the hash function H' is used to hash the public key PK_i of a user U_i to an domain input $u_i = H'(PK_i)$ used in the threshold secret sharing scheme. Hence, it is required H' to be a collision resistant hash function. Alternatively, instead of using a hash function H', a user U_i can select its own unique u_i and then register along with its public key PK_i to a certification authority. Thus each user U_i has a unique public key PK_i associated with a unique public value u_i. For the system simplicity, we use the hash function H'.

$$\mathbb{G}_1 = \langle g \rangle, \quad e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2, \quad H : \{0,1\}^* \to \mathbb{G}_1, \quad H' : \mathbb{G}_1 \to \mathbb{Z}_q.$$

return $\mathsf{param} = (q, \mathbb{G}_1, \mathbb{G}_2, g, e, H, H')$.

– $(\mathsf{PK}_i, \mathsf{SK}_i) \leftarrow \mathsf{KeyGen}(\mathsf{param})$: With the system wide parameters param, each user U_i randomly chooses a secret key $\mathsf{SK}_i = x_i \in_R \mathbb{Z}_q$. Then its public key can be computed as $\mathsf{PK}_i = g^{x_i}$.

$$\mathsf{SK}_i = x_i \in \mathbb{Z}_q, \quad \mathsf{PK}_i = g^{x_i}.$$

return $(\mathsf{PK}_i, \mathsf{SK}_i)$.

– $C \leftarrow \mathsf{TBEKS}(\{\mathsf{PK}_1, \ldots, \mathsf{PK}_n\}, t, W)$: To encrypt a keyword W, the data owner first obtains all the public keys of the target users $S = \{U_1, \ldots, U_n\}$. Then the data owner obtains the associated input values $\mathcal{U} = \{u_i = H'(\mathsf{PK}_i) \mid U_i \in S\}$. Having n input values u_i and $\mathsf{PK}_i = g^{x_i}$, we can recover/construct a polynomial $g^{p(u_j)} = \mathsf{PK}_i^{\sum_{U_i \in S} \Delta_{ij}^S}$. To form an $(n, 2n-t)$ threshold secret sharing scheme, the data owner chooses $n-t$ unique domain input values $\mathcal{D} \leftarrow \mathbb{Z}_q^{n-t}$, where $\mathcal{U} \cap \mathcal{D} = \emptyset$, for $n-t$ dummy users. Given the $n-t$ dummy users as a dummy user set D, the data owner can compute the public keys of the dummy users by computing $\mathsf{PK}_j = g^{p(u_j)}$ for all $u_j \in \mathcal{D}$. The detail of this algorithm works as follows.

$$Q = g^{p(0)} = \mathsf{PK}_i^{\sum_{U_i \in S} \Delta_{i0}^S}, \quad s \in_R \mathbb{Z}_q, \quad C_1 = g^s, \quad C_2 = e(H(W), Q)^s,$$

For each dummy user $U_j \in D$,

$$\mathsf{PK}_j = g^{p(u_j)} = \mathsf{PK}_i^{\sum_{U_i \in S} \Delta_{ij}^S}, \quad K_j = e(H(W)^s, \mathsf{PK}_j).$$

return $C = (S, t, D, C_1, C_2, \{K_j\}_{U_j \in D})$.

To improve computational efficiency, it is possible to reuse Q and PK_j for the same user set S and different keywords, since these two variables are

irrelevant to the keyword W and the randomness s. When calculating K_j, we can calculate $H(W)^s$ before the loop so that all we need is a pairing operation. For the ciphertext C, it is not necessary to include all the public keys and the associated domain input values for both real users and dummy users since we only need the domain input values later. For better efficiency, this algorithm can return the ciphertext as $C = (\mathcal{U}, t, \mathcal{D}, C_1, C_2, \{K_j\}_{U_j \in D})$. As the values in $\mathcal{D}$ are not required to be chosen uniformly, the data owner can choose a continuous interval that $\mathcal{D} = \{r, r+1, r+2, \ldots, r+n-t-1\}$ where $r \in_R \mathbb{Z}_q$. Thus $\mathcal{D}$ can be represented in two numbers in $\mathbb{Z}_q$. Hence, for the best result, the ciphertext size is $(n+3)\mathbb{Z}_q + (n-t+2)\mathbb{G}_1$.

– $T \leftarrow \mathsf{Trapdoor}(\mathsf{SK}_i, W)$: The user U_i generates the trapdoor T for the keyword W simply using its secret key. Then the user U_i uploads the trapdoor to the server via a secure communication channel.

$$T = H(W)^{x_i}.$$

return T.

– $1/0 \leftarrow \mathsf{Test}(\{T_1, \ldots, T_t\}, C)$: Upon receiving t trapdoors from the users $A = \{U_1, \ldots, U_t\}$ where $|A \subset S| = t$, the server can run the following algorithm. If more than t target trapdoors are uploaded, the server only picks the first t trapdoors.

$$\begin{aligned} &\text{For each user } U_i \in A, \quad K_i = e(T_i, C_1), \\ &B = A \cup D, \quad K = \prod_{U_i \in B} K_i^{\Delta_{i0}^B}. \end{aligned}$$

return $K \stackrel{?}{=} C_2$.

Theorem 4.1. *The proposed TBEKS scheme is correct.*

Proof. Correctness is verified as following. First, K_i can be calculated as

$$K_i = e(H(W)^s, \mathsf{PK}_i) = e(H(W)^s, g^{x_i}) = e(H(W)^{x_i}, g^s) = e(T_i, C_1).$$

Then we continue verify the correctness of K,

$$K = \prod_{U_i \in B} K_i^{\Delta_{i0}^B} = \prod_{U_i \in B} e(H(W)^s, \mathsf{PK}_i)^{\Delta_{i0}^B} = e(H(W), \mathsf{PK}_i^{\sum_{U_i \in B} \Delta_{i0}^B})^s.$$

Since the $(n, 2n-t)$ threshold secret sharing scheme is constructed by n real users S, distributing shares to $n-t$ dummy users D, any n users in $S \cup D$ can recover the

polynomial p used in the $TBEKS$ algorithm. Having $|D| = n-t$ and $S \cap D = \emptyset$ and $|A \subset S| = t$, we conclude that $A \cap D = \emptyset$ and further $|B = (A \cup D) \subset (S \cup D)| = n$. Thus the algorithm can recover the polynomial p with the users B. Then we have,

$$\mathsf{PK}_i^{\sum_{U_i \in B} \Delta_{i0}^B} = g^{p(0)} = Q.$$

Finally,

$$K = e(H(W), Q)^s = C_2.$$

□

4.4 Security Proof

Theorem 4.2. *The proposed TBEKS scheme is IND-T-CKA secure. If an adversary $\mathcal{A}$ can win the IND-T-CKA game (Fig. 4.1) with the advantage ε, an algorithm $\mathcal{S}$ can be constructed to solve Decisional Bilinear Diffie-Hellman (DBDH) problem in polynomial time with the advantage $\varepsilon' \geq \frac{\varepsilon}{2e^2(q_C+1)(q_T+1)}$, querying $\mathcal{O}_{\mathsf{Corrupt}}$ for at most q_C times and $\mathcal{O}_{\mathsf{Trapdoor}}$ for at most q_T times.*

Proof. Let $\delta = (g, g^a, g^b, g^c, Z)$ be an instance of DBDH problem (recall Definition 2.11) that a simulator $\mathcal{S}$ is challenged to distinguish that $\delta \in \mathcal{D}_{\text{DBDH}}$ or $\delta \in \mathcal{D}_R$. From the DBDH instance $\mathbb{D}$, the simulator $\mathcal{S}$ is also given two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of the same order q, a generator g of $\mathbb{G}_1$ and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. The simulator $\mathcal{S}$ further choose two hash functions $H : \{0,1\}^* \to \mathbb{G}_1$ and $H' : \mathbb{G}_1 \to \mathbb{Z}_q$. Then the simulator $\mathcal{S}$ packs those parameters as $\mathsf{param} = (q, \mathbb{G}_1, \mathbb{G}_2, g, e, H, H')$ and passes param to the adversary $\mathcal{A}$.

At the same time, the simulator $\mathcal{S}$ simulates the three oracles as follows.

– $\mathcal{O}_H$: The hash function H is viewed as a random oracle for the adversary $\mathcal{A}$ simulated by the simulator $\mathcal{S}$. Upon requesting the hash value of the keyword W_i, the simulator $\mathcal{S}$ randomly tosses a coin $c_i \in \{0,1\}$ such that $\Pr[c_i = 0] = \alpha$ where α is determined later. The simulator $\mathcal{S}$ also chooses a random value $a_i \in_R \mathbb{Z}_q$. Then the simulator $\mathcal{S}$ computes the hash value h_i as

$$h_i = \begin{cases} g^{a \cdot a_i} & \text{if } c_i = 0, \\ g^{a_i} & \text{if } c_i = 1. \end{cases}$$

The distribution of $\{h_i\}$ is indistinguishable with a random distribution of $\mathbb{G}_1$. After that, the simulator $\mathcal{S}$ returns h_i to the adversary $\mathcal{A}$. In addition, the simulator $\mathcal{S}$ maintains a hash list $\mathcal{H} = \{W_i, c_i, a_i.h_i\}$. If the requested keyword W is on the list $\mathcal{H}$, the simulator $\mathcal{S}$ returns h_i directly.

– $\mathcal{O}_{\mathsf{KeyGen}}$: To create a user U_i, the simulator $\mathcal{S}$ randomly tosses a coin $d_i \in \{0,1\}$ such that $\Pr[d_i = 0] = \beta$ where β is determined later. The simulator also chooses a random value $x_i \in \mathbb{Z}_q$. Then the simulator $\mathcal{S}$ computes the secret key SK_i and the public key PK_i as follows.

$$\mathsf{SK}_i = \begin{cases} \text{unknown} & \text{if } d_i = 0, \\ x_i & \text{if } d_i = 1. \end{cases} \qquad \mathsf{PK}_i = \begin{cases} g^{b \cdot x_i} & \text{if } d_i = 0, \\ g^{x_i} & \text{if } d_i = 1. \end{cases}$$

In the case of $d_i = 0$, the secret key $\mathsf{SK}_i = b \cdot x_i$ cannot be computed by and is unknown to the simulator $\mathcal{S}$ since it is computational hard to compute b from g^b. The distribution of $\{\mathsf{PK}_i\}$ is indistinguishable with a random distribution of $\mathbb{G}_1$. After that, the simulator $\mathcal{S}$ returns PK_i to the adversary $\mathcal{A}$. In addition, the simulator $\mathcal{S}$ maintains a user key list $\mathcal{K} = \{U_i, d_i, \mathsf{SK}_i, \mathsf{PK}_i\}$.

– $\mathcal{O}_{\mathsf{Corrupt}}$: Upon requesting the secret key SK_i of a created user U_i, the simulator $\mathcal{S}$ searches the user key list $\mathcal{K}$ and checks the corresponding d_i value. If $d_i = 0$, the simulator $\mathcal{S}$ **aborts** since the secret key SK_i is unknown to $\mathcal{S}$. Otherwise, the simulator $\mathcal{S}$ returns SK_i to the adversary $\mathcal{A}$.

– $\mathcal{O}_{\mathsf{Trapdoor}}$: To create a created user U_i's trapdoor of a keyword W_j, the simulator $\mathcal{S}$ first looks up the hash list $\mathcal{H}$ for W_j. If $c_j = 0$, the simulator $\mathcal{S}$ simply **aborts**. Otherwise, the simulator $\mathcal{S}$ computes the trapdoor $T = \mathsf{PK}_i^{a_j}$ and returns it to the adversary $\mathcal{A}$. The correctness is verified as follows.

$$T = \mathsf{PK}_i^{a_j} = \begin{cases} g^{bx_ia_j} = h_i^{x_ib} = H(W_j)^{\mathsf{SK}_i} & \text{if } d_i = 0, \\ g^{x_ia_j} = h_i^{x_i} = H(W_j)^{\mathsf{SK}_i} & \text{if } d_i = 1. \end{cases}$$

Although the trapdoor is still able to be simulated in the case of $c_j = 1 \wedge d_i = 0$ as $T = H(W_j)^{\mathsf{SK}_i} = h_j^{x_i} = g^{a \cdot a_j x_i}$, the simulator $\mathcal{S}$ still aborts for the simplicity of this proof. In other words, the probability ε' of solving the DBDH problem is greater if the simulator $\mathcal{S}$ does not abort in the above case but makes the proof harder. As long as ε' is not negligible, it is still acceptable.

At some point, the adversary $\mathcal{A}$ outputs a target user set S, a target threshold value t and two target keyword W_0 and W_1. The simulator looks up the hash list $\mathcal{H}$ for W_0 and W_1. If the keyword is not on the list $\mathcal{H}$, the simulator asks the $\mathcal{O}_H$ oracle for its hash value and then the keyword is on the list. If the corresponding values c_0 and c_1 of the keywords W_0 and W_1 is equals to 1, the simulator $\mathcal{S}$ **aborts**. Otherwise, the simulator $\mathcal{S}$ randomly picks $b \in \{0,1\}$ such that $c_b = 0$. If there is only one $c = 0$, the simulator $\mathcal{S}$ has no choice and the value b is fixed. Then we have $H(W_b) = g^{a \cdot a_b}$. Due to the restrictions to the adversary $\mathcal{A}$, at least one user

U_σ in S has not been corrupted. The simulator $\mathcal{S}$ looks up the user key list $\mathcal{K}$ for that user. If the corresponding value $d_\sigma = 1$, the simulator $\mathcal{S}$ **aborts**. Otherwise, the simulator $\mathcal{S}$ divides the user set S into two sets $S_0 = \{U_i \in S \mid d_i = 0\}$ and $S_1 = \{U_i \in S \mid d_i = 1\}$. Intuitively, $S_0 \neq \emptyset$ because of the existence of U_σ. After that, the simulator $\mathcal{S}$ sets $C_1 = g^c$. Before simulating C_2, we first seek how the genuine ciphertext is computed:

$$\begin{aligned} C_2 &= e(H(W), Q)^c = e(H(W), \mathsf{PK}_i^{\sum_{U_i \in S} \Delta_{i0}^S})^c \\ &= e(H(W), \mathsf{PK}_i^{\sum_{U_i \in S_0} \Delta_{i0}^S})^c \cdot e(H(W), \mathsf{PK}_i^{\sum_{U_i \in S_1} \Delta_{i0}^S})^c \\ &= e(g^{a \cdot a_b}, g^{b \cdot \sum_{U_i \in S_0} \Delta_{i0}^S x_i})^c \cdot e(g^{a \cdot a_b}, g^{\sum_{U_i \in S_1} \Delta_{i0}^S x_i})^c \\ &= e(g, g)^{abc \cdot a_b \sum_{U_i \in S_0} \Delta_{i0}^S x_i} \cdot e(g^a, g^c)^{a_b \sum_{U_i \in S_1} \Delta_{i0}^S x_i}. \end{aligned}$$

The simulator $\mathcal{S}$ replaces the $e(g, g)^{abc}$ part with Z and sets C_2 as

$$C_2 = Z^{a_b \sum_{U_i \in S_0} \Delta_{i0}^S x_i} \cdot e(g^a, g^c)^{a_b \sum_{U_i \in S_1} \Delta_{i0}^S x_i}.$$

After that, the simulator $\mathcal{S}$ selects a set D of $n - t$ dummy users with the same restrictions in the normal construction (i.e. $\mathcal{U} \cap \mathcal{D} = \emptyset$). Similar to C_2, the simulator $\mathcal{S}$ computes K_j as

$$K_j = Z^{a_b \sum_{U_i \in S_0} \Delta_{ij}^S x_i} \cdot e(g^a, g^c)^{a_b \sum_{U_i \in S_1} \Delta_{ij}^S x_i}.$$

Finally, the simulator $\mathcal{S}$ packs the ciphertext $C = (S, t, D, C_1, C_2, \{K_j\}_{U_j \in D})$ and sends to the adversary $\mathcal{A}$. Note that the resulted ciphertext C is consistent if and only if $Z = e(g, g)^{abc}$.

Eventually, the adversary $\mathcal{A}$ outputs a guess b'. If $b = b'$, it means the ciphertext is consistent and it is believed that $Z = e(g, g)^{abc}$. Hence, the simulator $\mathcal{S}$ outputs $\delta \in \mathcal{D}_{\text{DBDH}}$. Otherwise, the simulator $\mathcal{S}$ outputs $\delta \in \mathcal{D}_R$.

Lemma 4.1. *Let ρ be the probability of the simulator $\mathcal{S}$ not aborting. The advantage ε' of the simulator $\mathcal{S}$ solving the DBDH problem is at least $\frac{\rho\varepsilon}{2}$, assuming the probability of $\delta \in \mathcal{D}_{DBDH}$ is $\frac{1}{2}$ and the adversary $\mathcal{A}$ wins the IND-T-CKA game (Fig. 4.1) with the advantage ε.*

Proof. We prove this lemma by calculating the probability of the simulator $\mathcal{S}$ succeeding. If $\delta \in \mathcal{D}_R$, the behaviour of the adversary $\mathcal{A}$ is unpredictable. Thus, the simulator $\mathcal{A}$ succeeds at least better than a random guess with succeeding probability of $\frac{1}{2}$. Similarly, if the simulator S aborts, we just have a random guess. Otherwise, we take the result of the adversary $\mathcal{A}$ with the correct probability of $\frac{1}{2} + \varepsilon$. Let E_R be the event that the simulator $\mathcal{S}$ succeeds with a random guess. We

have,

$$\begin{aligned}
&\Pr[\mathcal{S} \text{ succeeds}] \\
&= \frac{1}{2}\Pr[\mathcal{S} \text{ succeeds} \mid \delta \in \mathcal{D}_{DBDH}] + \frac{1}{2}\Pr[\mathcal{S} \text{ succeeds} \mid \delta \in \mathcal{D}_R] \\
&\geq \frac{1}{2}\left(\Pr[\mathcal{S} \text{ does not abort}] \cdot \Pr[\mathcal{A} \text{ succeeds}] + \Pr[\mathcal{S} \text{ aborts}] \cdot \Pr[E_R]\right) + \frac{1}{2}\Pr[E_R] \\
&= \frac{1}{2}\left(\Pr[\mathcal{S} \text{ does not abort}] \cdot (\frac{1}{2} + \varepsilon) + \Pr[\mathcal{S} \text{ aborts}] \cdot \frac{1}{2}\right) + \frac{1}{2} \cdot \frac{1}{2} \\
&= \frac{1}{2}\left(\rho \cdot (\frac{1}{2} + \varepsilon) + (1-\rho) \cdot \frac{1}{2}\right) + \frac{1}{2} \cdot \frac{1}{2} \\
&= \frac{1}{2}\rho\varepsilon + \frac{1}{2}.
\end{aligned}$$

Since $\Pr[\mathcal{S} \text{ succeeds}] \geq \frac{1}{2}\rho\varepsilon + \frac{1}{2}$ and $\mathsf{Adv}_{\mathcal{S}}^{\mathrm{DBDH}} = \left|\mathcal{S} \text{ succeeds}] - \frac{1}{2}\right|$, we have

$$\mathsf{Adv}_{\mathcal{S}}^{\mathrm{DBDH}} \geq \frac{\rho\varepsilon}{2}.$$

□

Lemma 4.2. *The simulator $\mathcal{S}$ does not abort with the probability ρ at least $\frac{1}{e^2(q_C+1)(q_T+1)}$, querying $\mathcal{O}_{\mathsf{Corrupt}}$ for at most q_C times and $\mathcal{O}_{\mathsf{Trapdoor}}$ for at most q_T times.*

Proof. There are 4 possible points that the simulator $\mathcal{S}$ may abort.

1. The simulator $\mathcal{S}$ aborts in answering $\mathcal{O}_{\mathsf{Corrupt}}$ queries if $d_i = 0$. The single abort probability is β. The probability of not aborting for all $\mathcal{O}_{\mathsf{Corrupt}}$ queries is $\Pr[E_1] = (1-\beta)^{q_C}$.

2. The simulator $\mathcal{S}$ aborts in answering $\mathcal{O}_{\mathsf{Trapdoor}}$ queries if $c_i = 0$. The single abort probability is α. The probability of not aborting for all $\mathcal{O}_{\mathsf{Trapdoor}}$ queries is $\Pr[E_2] = (1-\alpha)^{q_T}$.

3. The simulator $\mathcal{S}$ aborts in the challenge phase if $c_0 = c_1 = 1$. The probability of not aborting for this event is $\Pr[E_3] = 1 - (1-\alpha)^2 = 2\alpha - \alpha^2$. Since $\alpha \in [0,1]$, we have

$$\alpha \leq 1 \implies \alpha^2 \leq \alpha \implies 0 \leq \alpha - \alpha^2 \implies \alpha \leq 2\alpha - \alpha^2 = \Pr[E_3].$$

4. The simulator $\mathcal{S}$ aborts in the challenge phase if $d_\delta = 1$. The probability of not aborting for this event is $\Pr[E_4] = \beta$.

Since all the events are independent, we have

$$\rho = \Pr[E_1] \cdot \Pr[E_2] \cdot \Pr[E_3] \cdot \Pr[E_4] \geq \alpha(1-\alpha)^{q_T}\beta(1-\beta)^{q_C}.$$

The function $\alpha(1-\alpha)^{q_T}\beta(1-\beta)^{q_C}$ is maximised when $\alpha = \frac{1}{q_T+1}$ and $\beta = \frac{1}{q_C+1}$. Now we have

$$\begin{aligned}\rho &\geq \frac{1}{q_T+1}(1-\frac{1}{q_T+1})^{q_T} \cdot \frac{1}{q_C+1}(1-\frac{1}{q_C+1})^{q_C} \\ &\geq \frac{1}{q_T+1}\left(\lim_{q_T\to\infty}(1-\frac{1}{q_T+1})^{q_T}\right) \cdot \frac{1}{q_C+1}\left(\lim_{q_C\to\infty}(1-\frac{1}{q_C+1})^{q_C}\right) \\ &= \frac{1}{e^2(q_C+1)(q_T+1)}.\end{aligned}$$

Note that the statement $(1-\frac{1}{q_T+1})^{q_T} \geq \frac{1}{e} = \lim_{q_T\to\infty}(1-\frac{1}{q_T+1})^{q_T}$ is always true for any q_T. Similar statement for q_C also applies. □

Combining $\mathsf{Adv}_{\mathcal{S}}^{\mathrm{DBDH}} \geq \frac{\rho\varepsilon}{2}$ from Lemma 4.1 and $\rho \geq \frac{1}{e^2(q_C+1)(q_T+1)}$ from Lemma 4.2, we have

$$\varepsilon' = \mathsf{Adv}_{\mathcal{S}}^{\mathrm{DBDH}} \geq \frac{\varepsilon}{2e^2(q_C+1)(q_T+1)}.$$

□

4.5 Chapter Summary

In this chapter, we defined Threshold Broadcast Encryption with Keyword Search (TBEKS) scheme and its IND-T-CKA security model. We proposed the first TBEKS scheme and proved it is IND-T-CKA secure, assuming the hardness of the DBDH problem. In our TBEKS scheme, we consider the server to be honest but curious and we do not allow the server to collude with the users. It is an open problem to build a scheme that is secure against a malicious server that may collude with other users.

Chapter 5

Linear Encryption with Keyword Search

Nowadays an increasing amount of data stored in the public cloud need to be searched remotely for fast accessing. For the sake of privacy, the remote files are usually encrypted, which makes them difficult to be searched by remote servers. It is also harder to efficiently share encrypted data in the cloud than those in plaintext. In this chapter, we develop a searchable encryption framework called Linear Encryption with Keyword Search (LEKS) that can semi-generically convert some existing encryption schemes meeting our Linear Encryption Template (LET) to be searchable without re-encrypting all the data. For allowing easy data sharing, we convert a Key-Policy Attribute-Based Encryption (KP-ABE) scheme to a Key-Policy Attribute-Based Encryption with Keyword Search (KP-ABKS) scheme as a concrete instance of our LEKS framework, making both the encrypted data and the search functionality under fine-grained access control. Notably, the resulting KP-ABKS is the first proven secure Attribute-Based Encryption with Keyword Search (ABKS) scheme with the security of Indistinguishability under Selective-ID Adaptive Chosen Keyword Attacks (IND-sCKA) in the random oracle model, assuming the hardness of the Decisional ℓ-Combined Bilinear Diffie-Hellman (ℓ-DCBDH) problem derived from the Decisional Bilinear (P, f)-Diffie-Hellman ((P, f)-DBDH) problem family. Parts of this work appeared in [ZYM16].

5.1 Introduction

Cloud computing [MG11] provides on-demand computing resources that are accessible via the Internet, including computing power and data storage. With the convenient cloud services, users can outsource their computing resources to the cloud, and access them through terminals with low computing capabilities, such as mobile devices. Usually, those terminals also have low network connectivity due to the transmission technology, signal strength, access cost, and other factors.

In terms of data storage, one important function for cloud data storage is data search. Since all the user data are stored on the cloud server, users have to send search queries to the server to search for the data containing certain keywords.

However, the normal search operation for plaintext is no longer working when data privacy is considered, since all the data are encrypted and cannot be read by the server.

To perform search on encrypted data, it is impractical for the user to do the search locally with all the data downloaded from the server, due to the high demand on the bandwidth. It is also impractical to give the server the user secret key due to privacy concerns. Thus searchable encryption has been introduced such that the search operation is performed by the server, but the server cannot get any meaningful information from the search query or the encrypted data. In searchable encryption, all the data files and their associated keywords are encrypted. To search for the data with certain keyword, the user generates a trapdoor for the keyword and enquires the server with the trapdoor. The server searches the whole database to locate the data where the encrypted keyword matches the keyword embedded in the trapdoor. During the searching process, the server only knows whether an encrypted keyword matches the user trapdoor or not, and nothing else. After that, the server returns the search result to the user who can download the ciphertexts and decrypt the data.

In Public-key Encryption with Keyword Search (PEKS) [BDCOP04], the data and the keywords are encrypted for only one user (i.e., the intended receiver of the data). Hence, only that particular user can do the search and decryption. In contrast, in Attribute-Based Encryption (ABE) [SW05], data can be encrypted with certain attributes. For instance, Alice can encrypt some data with attributes "full-time" and "student". Later, any user can decrypt the resulting ciphertext if the attributes in the ciphertext match the policy associated with the user. Thus Bob associated with a policy "(full time AND student) OR staff" can decrypt the above ciphertext. The corresponding searchable encryption for ABE is named Attribute-Based Encryption with Keyword Search (ABKS) [SYL+14, ZXA14]. As in ABE, Alice can now encrypt the data and its associated keywords using certain attributes. After uploading the ciphertexts to the server, Bob can do the search and decryption since the attributes used by Alice in the encryption matches Bob's policy. This feature is very important in the cloud environment where a user can share data with multiple users by encrypting the data only once. However, to the best of our knowledge, no ABKS scheme proposed in the literature is proven secure. Hence, one of our goals is to construct ABKS schemes with provable security.

In addition, keyword search functionality is usually associated with an encryption scheme where both the data and the keywords are encrypted for the same receiver(s). This chapter also aims to provide a universal construction of searchable encryption schemes from some existing encryption schemes. This enables us to add a compatible keyword search functionality to an existing cryptosystem without

re-encrypting all the data.

5.1.1 Related Work

Diffie and Hellman introduced the notion of Public Key Encryption (PKE) [DH76] where Alice encrypts a message with Bob's public key, and Bob decrypts the ciphertext with his secret key. Based on the idea of using the user identity as the public key [Sha85], Boneh and Franklin proposed a practical Identity-Based Encryption (IBE) scheme [BF01] where Alice encrypts the message with Bob's identity. In 2005, Sahai and Waters introduced Fuzzy Identity-Based Encryption which can be treated as the first ABE [SW05], an instance of Functional Encryption (FE) [BSW11]. In ABE, the decryption keys of the users and the ciphertexts are associated with access policies and attributes, respectively. If and only if the attributes match the policy, the ciphertext can be successfully decrypted. Depending on how the identity and the ciphertext are associated, ABE schemes are classified into Key-Policy Attribute-Based Encryption (KP-ABE) [ALdP11, GPSW06, PTMW06, SW05] and Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [BSW07]. In KP-ABE, Bob's secret key is associated with a policy. After receiving the ciphertext encrypted with some attributes from Alice, Bob can decrypt it if and only if the attributes match his policy. In CP-ABE, the ciphertexts are associated with policies, and the secret keys are associated with attributes.

To enable the search functionality for encrypted data, various searchable encryption schemes [ABC$^+$08, BDCOP04, BW06, HL07, SYL$^+$14, ZXA14] have been proposed under different settings. Boneh et al. [BDCOP04] introduced Public-key Encryption with Keyword Search (PEKS), which is used with a conventional public key encryption scheme. Later, Identity-Based Searchable Encryption (IBKS) schemes were also proposed [ABC$^+$08, BW06]. Recently, due to the popularity of ABE, there have been some research works on Attribute-Based Encryption with Keyword Search (ABKS) [SYL$^+$14, ZXA14]. In addition, there are also keyword search schemes for other encryption variants, such as Broadcast Encryption (BE) [AFI06].

To the best of our knowledge, [SYL$^+$14, ZXA14] are the only ABKS schemes proposed in the literature. However, neither of those schemes is proven secure. In particular, after analysing the ABKS scheme in [SYL$^+$14], we found the scheme is flawed (details are given in Appendix A) where a malicious adversary can always distinguish keywords from a ciphertext. For the Key-Policy Attribute-Based Encryption with Keyword Search (KP-ABKS) scheme in [ZXA14], the security proof is not valid (details are given in Section 5.7.3) and thus the security of this scheme remains unknown. For the Ciphertext-Policy Attribute-Based Encryption with Key-

word Search (CP-ABKS) scheme in [ZXA14], no formal security proof has been provided.

In terms of provable security, it depends on the hardness of some computational problems (e.g. Discrete Logarithm (DL) problem, Computational Diffie-Hellman (CDH) problem, Decisional Diffie-Hellman (DDH) problem [DH76], etc.). Shoup [Sho97] introduced the generic group model which was used to obtain the complexity lower bound regarding the hardness of DL, CDH, and DDH problems. Later, in order to deal with bilinear maps, Boneh et al. [BBG05] introduced the generic bilinear group model and the general Diffie-Hellman Exponent Problem. Besides, the generic bilinear group model is also used in [BB07, BBS04] for analysing the Decision Linear (DLIN) problem and q-Strong Diffie-Hellman (q-SDH) problem.

5.1.2 Our Contribution

In this chapter, we introduce a new problem family named Decisional Bilinear (P, f)-Diffie-Hellman ((P, f)-DBDH). We prove the (P, f)-DBDH problem is computationally hard in generic bilinear group model if the polynomial f is not dependent on the polynomial set P. Based on the (P, f)-DBDH problem, we derive a hard computational problem named Decisional ℓ-Combined Bilinear Diffie-Hellman (ℓ-DCBDH) problem.

As the main contribution of this work, we introduce two new notions named Linear Encryption Template (LET) and Linear Encryption with Keyword Search (LEKS), and provide their formal definitions. LET can model different asymmetric encryption schemes, including but not limited to PKE schemes, IBE schemes and ABE schemes, which have the property of linearity. The linearity property requires a sub-algorithm $e(g, g)^{\alpha s} \leftarrow \mathcal{D}(\mathsf{SK}, C_1, \dots)$ in the decryption algorithm where SK is the secret key involved, $(C_1, \dots)$ are the ciphertext components and for all $t \in \mathbb{Z}_p$, $\mathcal{D}(\mathsf{SK}^t, C_1, \dots) = D(\mathsf{SK}, C_1, \dots)^t$. Given an encryption fitting LET, we provide a semi-generic conversion from the encryption scheme to a LEKS scheme where the construction is generic but we require security proofs for individual conversions. We also define two security models for LEKS schemes: Indistinguishability under Selective-ID Adaptive Chosen Keyword Attacks (IND-sCKA) and Indistinguishability under Adaptive Chosen Keyword Attacks (IND-CKA). With LET and our conversion from LET to LEKS, we can construct PEKS from PKE, IBKS from IBE, ABKS from ABE, and so on.

To illustrate the feasibility of our semi-generic framework, we give an instance of LET and then apply our conversion to produce a LEKS scheme. We first show that a variant [PTMW06] of Goyal et al.'s ABE scheme [GPSW06] fits LET by proving it has the property of linearity. Then we apply our LEKS conversion to convert

the KP-ABE scheme into a KP-ABKS scheme. After that, we prove the resulting KP-ABKS scheme is IND-sCKA secure in the random oracle model, assuming the hardness of the ℓ-DCBDH problem. It is worth noting that to the best of our knowledge, our KP-ABKS scheme converted from the KP-ABE scheme is the first proven secure KP-ABKS scheme.

5.1.3 Chapter Organisation

The rest of this chapter is organised as follows. Beginning with Section 5.2, we define (P, f)-DBDH problem family and ℓ-DCBDH problem, and prove the hardness of those problems. In Section 5.3, we define LEKS and its security model, followed by the definition of LET and the LEKS conversion from LET. At a glance, we show a simple LEKS conversion that converts a PKE scheme to a PEKS scheme in Section 5.6. After that, an instance of LEKS conversion is given in Section 5.7, converting a KP-ABE scheme to a KP-ABKS scheme. The resulted KP-ABKS scheme is proven secure in Section 5.7.3 under the security model defined in Section 5.3.2. Finally, this chapter is summarised in Section 5.8.

5.2 Decisional Diffie-Hellman Problem Family

There is a family of Diffie-Hellman problems, which could be used for constructing our cryptographic schemes.

Definition 5.1. *Let $P = (p_1, \ldots, p_s) \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be a s-tuples of n-variate polynomial over $\mathbb{F}_p$. We define that a polynomial $f \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ is dependent on P if exists $s^2 + 2s$ constants $a_{i,j}$, b_k and c_l such that*

$$f = \frac{\sum_{i=1}^{s}\sum_{j=1}^{s} a_{i,j}p_ip_j}{\sum_{k=1}^{s} b_kp_k} + \sum_{l=1}^{s} c_lp_l \quad or \quad f = \sum_{l=1}^{s} c_lp_l \pm \sqrt{\sum_{i=1}^{s}\sum_{j=1}^{s} a_{i,j}p_ip_j}$$

Equivalently, f is dependent on P if exists $s^2 + s + 1$ constants $a_{i,j}$, b_k and c

$$cf^2 + \sum_{k=1}^{s} b_kp_kf + \sum_{i=1}^{s}\sum_{j=1}^{s} a_{i,j}p_ip_j = 0$$

where at least one of b_k or c is non-zero.

Let $g^{P(x_1,\ldots,x_n)} = (g^{p_1(x_1,\ldots,x_n)}, \ldots, g^{p_s(x_1,\ldots,x_n)})$, d_f denote the total degree of $f \in \mathbb{F}_p[X_1, \ldots, X_n]$, and $d_P = \max\{d_f \mid f \in P \in \mathbb{F}_p[X_1, \ldots, X_n]^s\}$. We present the family of Diffie-Hellman problems as follows.

Definition 5.2 (Decisional Bilinear (P, f)-Diffie-Hellman problem)**.** *Let $P = (p_1, \ldots, p_s) \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be a s-tuples of n-variate polynomial over $\mathbb{F}_p$, $f \in \mathbb{F}_p[X_1, \ldots, X_n]$ be a n-variate polynomial over $\mathbb{F}_p$. Let $(x_1, \ldots, x_n)$ be uniformly and independently chosen from $\mathbb{Z}_p$, and Z be uniformly and independently chosen from $\mathbb{G}_1$. Giving two probability distributions $\mathcal{D}_{\text{target}} = (g^{P(x_1,\ldots x_n)}, g^{f(x_1,\ldots x_n)})$ and $\mathcal{D}_{\text{rand}} = (g^{P(x_1,\ldots x_n)}, Z)$, there is an algorithm $\mathcal{A}$ can distinguish $\mathcal{D}_{\text{target}}$ and $\mathcal{D}_{\text{rand}}$ with advantage:*

$$\mathsf{Adv}_{\mathcal{A}}^{(P,f)-\text{DBDH}} = \frac{1}{2}\left|\Pr\left[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_{\text{target}})\right] - \Pr\left[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_{\text{rand}})\right]\right|$$

where $D \in_R \mathcal{D}$ represents that D is uniformly and independently chosen from $\mathcal{D}$. Alternatively, the problem can be represented as

$$b \in_R \{0,1\}, \quad Z_b = g^{f(x_1,\ldots x_n)}, \quad Z_{1-b} \in_R \mathbb{G}_1,$$
$$\mathsf{Adv}_{\mathcal{A}}^{(P,f)-\text{DBDH}} = \left|\Pr\left[b = b' \leftarrow \mathcal{A}(g^{P(x_1,\ldots x_n)}, Z_0, Z_1)\right] - \frac{1}{2}\right|$$

As from the definition above, Decisional Bilinear (P, f)-Diffie-Hellman ((P, f)-DBDH) problem family is an enhanced DDH problem on the group $\mathbb{G}_1$ where the adversary $\mathcal{A}$ is now able to conduct bilinear pairing operations on $\mathbb{G}_1$. The (P, f)-DBDH problem family is computational hard if and only if the advantage $\mathsf{Adv}_{\mathcal{A}}^{(P,f)-\text{DBDH}}$ is negligible. Since there is no any known proof of the hardness of this problem family, we show the complexity lower bound in the generic bilinear group model [BBG05]. As in [BBG05], we emphasise that a lower bound in generic groups does not imply a lower bound in any specific group.

Theorem 5.1. *Let $\varepsilon_1, \varepsilon_2 : \mathbb{Z}_p^+ \to \{0,1\}^m$ be two random encodings (injective maps) where $\mathbb{G}_1 = \{\varepsilon_1(x) \mid x \in \mathbb{Z}_p^+\}$, $\mathbb{G}_2 = \{\varepsilon_2(x) \mid x \in \mathbb{Z}_p^+\}$. Let $d = 2 \cdot \max(d_P, d_f)$. If f is not dependent on P, the lower bound of the advantage $\mathsf{Adv}^{(P,f)-\text{DBDH}}$ of solving the (P, f)-Diffie-Hellman problem (Definition 5.2) for the adversary $\mathcal{A}$ is stated as follows with at most $q_{1,\times}$, $q_{2,\times}$ queries to the group operation oracles $\mathcal{O}_\times^1$, $\mathcal{O}_\times^2$ and q_e queries to the bilinear pairing oracle $\mathcal{O}_e : \varepsilon_1 \times \varepsilon_1 \to \varepsilon_2$.*

$$\mathsf{Adv}_{\mathcal{A}}^{(P,f)-\text{DBDH}} \leq \frac{(q_{1,\times} + q_{2,\times} + q_e + s + 2)^2 d}{2p}$$

Proof. Let $\mathcal{S}$ be the simulator that simulates the (P, f)-DBDH problem environment for the adversary $\mathcal{A}$. The simulator $\mathcal{S}$ maintains two polynomial-sized dynamic lists:

$$L_1 = \{(p_i, \varepsilon_{1,i})\}, \quad L_2 = \{(q_i, \varepsilon_{2,i})\},$$

where $i = 1$ to the size of the list, $p_i, q_i \in \mathbb{F}_p[X_1, \ldots X_n, Y_0, Y_1]$. Initially, the

simulator $\mathcal{B}$ uniformly chooses $\varepsilon_{1,i} \in_R \{0,1\}^m$ for each $p_i \in P$, and set all s pairs of $(p_i, \varepsilon_{1,i})$ as L_1. Then the simulator $\mathcal{B}$ sets $p_{s+1} = Y_1$, $p_{s+2} = Y_2$, and uniformly chooses $\varepsilon_{1,s+1}, \varepsilon_{1,s+2} \in_R \{0,1\}^m$. After adding the pairs $(p_{s+1}, \varepsilon_{1,s+1})$, $(p_{s+2}, \varepsilon_{1,s+2})$ to the list L_1, the two lists are initialised as

$$L_1 = \{(p_i, \varepsilon_{1,i})\}_{i=1\ldots s+2}, \quad L_2 = \emptyset.$$

Since the simulator $\mathcal{B}$ can increase m to increase the hardness of the adversary $\mathcal{A}$ guessing the elements in $\mathbb{G}_1$ and $\mathbb{G}_2$, we assume that the adversary $\mathcal{A}$ can only query the encoded strings obtained from the simulator $\mathcal{B}$.

At the beginning of the game from Definition 5.2, $\mathcal{B}$ sends $\{\varepsilon_{1,i}\}_{i=1\ldots s+2}$ to the adversary $\mathcal{A}$ as $(g^{P(x_1,\ldots x_n)}, Z_0, Z_1)$. After that, the simulator $\mathcal{B}$ simulates the group operation oracles $\mathcal{O}_\times^1$, $\mathcal{O}_\times^2$ and the bilinear pairing oracle $\mathcal{O}_e$ as follows.

- $\mathcal{O}_\times^\varphi$: The group operation involves two operands $\varepsilon_{\varphi,a}$ and $\varepsilon_{\varphi,b}$. Based on these operands, the simulator $\mathcal{B}$ searches the list L_φ for the corresponding polynomials p_a and p_b. Then the simulator $\mathcal{B}$ performs a polynomial addition $p_c = p_a + p_b$ for multiplication or a polynomial subtraction $p_c = p_a - p_b$ for division. If p_c is in the list L_φ, the simulator $\mathcal{B}$ returns the corresponding $\varepsilon_{\varphi,c}$ to the adversary $\mathcal{A}$. Otherwise, the simulator $\mathcal{B}$ uniformly chooses $\varepsilon_{\varphi,c} \in_R \{0,1\}^m$, where $\varepsilon_{\varphi,c}$ is unique among the encoding strings in L_φ, and appends the pair $(p_c, \varepsilon_{\varphi,c})$ into the list L_φ. Finally, the simulator $\mathcal{B}$ returns $\varepsilon_{\varphi,c}$ to the adversary $\mathcal{A}$.
- $\mathcal{O}_e$: The bilinear pairing operation involves two operands $\varepsilon_{1,a}$ and $\varepsilon_{1,b}$. Based on these operands, the simulator $\mathcal{B}$ searches the list L_1 for the corresponding polynomials p_a and p_b. Then the simulator $\mathcal{B}$ performs a polynomial multiplication $p_c = p_a \cdot p_b$. If p_c is in the list L_2, the simulator $\mathcal{B}$ returns the corresponding $\varepsilon_{2,c}$ to the adversary $\mathcal{A}$. Otherwise, the simulator $\mathcal{B}$ uniformly chooses $\varepsilon_{2,c} \in_R \{0,1\}^m$, where $\varepsilon_{2,c}$ is unique among the encoding strings in L_2, and appends the pair $(p_c, \varepsilon_{2,c})$ into the list L_2. Finally, the simulator $\mathcal{B}$ returns $\varepsilon_{2,c}$ to the adversary $\mathcal{A}$.

After querying at most $q_{1,\times}, q_{2,\times}, q_e$ times of corresponding oracles, the adversary $\mathcal{A}$ returns a bit b'. Finally, the simulator $\mathcal{B}$ chooses $x_1, \ldots, x_n, y \in_R \mathbb{F}_p$, $b \in_R \{0,1\}$, and sets $y_b = f(x_1, \ldots, x_n)$, $y_{1-b} = y$, $Y_0 = y_0$, $Y_1 = y_1$, $X_i = x_i$ for $i = 1, \ldots, n$. The simulation of $\mathcal{B}$ is perfect if the following statements hold:

$$\forall (p_i, \varepsilon_{1,i}), (p_j, \varepsilon_{1,j}) \in L_1 \wedge i \neq j, \quad p_i(x_1, \ldots x_n, y_0, y_1) \neq p_j(x_1, \ldots x_n, y_0, y_1)$$
$$\forall (q_i, \varepsilon_{2,i}), (q_j, \varepsilon_{2,j}) \in L_2 \wedge i \neq j, \quad q_i(x_1, \ldots x_n, y_0, y_1) \neq q_j(x_1, \ldots x_n, y_0, y_1)$$

If the above statements do not hold, the adversary $\mathcal{A}$ will find that some encoding strings are mapped from the same values, and thus deviate $\mathcal{B}$'s simulation from

the real oracles. Let the event **abort** be any statement above being false. The event **abort** occurs when $p_i - p_j = 0$, $q_i - q_j = 0$, $(p_i - p_j)(x_1, \ldots x_n, y_0, y_1) = 0$ or $(q_i - q_j)(x_1, \ldots x_n, y_0, y_1) = 0$. Hence, we bound the probability that the event **abort** occurs by analysing the following three cases:

1. $p_i - p_j = 0$: In this case, the polynomial $p_i - p_j$ falls into the form of

$$\sum_{k=1}^{s} a_k p_k + bY_0 + cY_1 \tag{5.1}$$

for some constants $a_k, b, c \in \mathbb{F}_p$ after group operations. Assuming that setting $Y_b = f(X_1, \ldots, X_n)$ makes $p_i - p_j = 0$, we have $\sum_{k=1}^{s} a_k p_k + b' Y_b = 0$ since Y_b does not involve y. In other words, Y_b is in the form of $\sum_{k=1}^{s} a'_k p_k$. By contraposition, if $f(X_1, \ldots, X_n)$ is not in the form of $\sum_{k=1}^{s} a'_k p_k$, then $p_i - p_j \neq 0$. As f is not dependent on P, which is not in the above form, this case never occurs.

2. $q_i - q_j = 0$: In this case, the polynomial $q_i - q_j$ falls into the form of

$$\sum_{k=1}^{s+2}\sum_{l=1}^{s+2} a_{k,l} p_k p_l = \sum_{k=1}^{s}\sum_{l=1}^{s} a_{k,l} p_k p_l + \sum_{u=1}^{s} b_u p_u Y_0 + \sum_{v=1}^{s} c_v p_v Y_1 + dY_0 Y_1 + eY_0^2 + fY_1^2$$

for some constants $a_{k,l}, b_u, c_v, d, e, f \in \mathbb{F}_p$ after group operations and bilinear pairings. Assuming that setting $Y_b = f(X_1, \ldots, X_n)$ makes $q_i - q_j = 0$, we have

$$\sum_{k=1}^{s}\sum_{l=1}^{s} a_{k,l} p_k p_l + \sum_{u=1}^{s} b'_u p_u Y_b + c' Y_b^2 = 0 \tag{5.2}$$

since Y_b does not involve y. In other words, Y_b is in the form of

$$\begin{cases} \dfrac{\sum_{k=1}^{s}\sum_{l=1}^{s} a_{k,l} p_k p_l}{\sum_{u=1}^{s} b'_u p_u} & \text{if } c' = 0, \\ \sum_{u=1}^{s} b''_u p_u \pm \sqrt{\sum_{k=1}^{s}\sum_{l=1}^{s} a'_{k,l} p_k p_l} & \text{otherwise.} \end{cases}$$

By contraposition, if $f(X_1, \ldots, X_n)$ is not in the above form, then $q_i - q_j \neq 0$. As f is not dependent on P, which is not in the above form, this case never occurs.

3. $(p_i - p_j)(x_1, \ldots x_n, y_0, y_1) = 0$ or $(q_i - q_j)(x_1, \ldots x_n, y_0, y_1) = 0$: Since we have set $Y_b = f(X_1, \ldots, X_n)$ and proved that it will not make $p_i - p_j = 0$ or

$q_i - q_j = 0$ in above two cases, we simplify this case as $(p_i - p_j)(x_1, \ldots x_n, y) = 0$ or $(q_i - q_j)(x_1, \ldots x_n, y) = 0$. By group operations and bilinear pairing operations (see Eqs. (5.1) and (5.2)), the maximum total degree of $(p_i - p_j)$ and $(q_i - q_j)$ is $d = 2 \cdot \max(d_P, d_f)$. Hence, the probability of randomly chosen $X_1, \ldots, X_n, Y_{1-b} \in_R \mathbb{F}_p$ being the root of $p_i - p_j$ or $q_i - q_j$, as shown in Eqs. (5.1) and (5.2), is at most $\frac{d}{p}$. As there are at most $2 \cdot \binom{q_{1,\times}+q_{2,\times}+q_e+s+2}{2}$ pairs of (p_i, p_j) and q_i, q_j, we have

$$\Pr[\text{abort}] \leq 2 \cdot \binom{q_{1,\times} + q_{2,\times} + q_e + s + 2}{2} \cdot \frac{d}{p} \leq \frac{(q_{1,\times} + q_{2,\times} + q_e + s + 2)^2 d}{p}.$$

If the event **abort** does not occur, the simulation by $\mathcal{B}$ is perfect. Therefore, the adversary $\mathcal{A}$ outputs b' with random guess with $\Pr[b = b' \mid \neg\text{abort}] = \frac{1}{2}$, since b is independent from $\mathcal{A}$'s view. Furthermore, we have

$$\begin{cases} \Pr[b = b'] \geq \Pr[b = b' \wedge \neg\text{abort}] \\ \Pr[b = b'] = \Pr[b = b' \wedge \neg\text{abort}] + \Pr[b = b' \wedge \text{abort}] \\ \qquad\qquad \leq \Pr[b = b' \wedge \neg\text{abort}] + \Pr[\text{abort}] \end{cases}$$

$$\begin{aligned} &\implies 0 \leq \Pr[b = b'] - \Pr[b = b' \wedge \neg\text{abort}] \leq \Pr[\text{abort}] \\ &\implies 0 \leq \Pr[b = b'] - \Pr[b = b' \mid \neg\text{abort}] \cdot \Pr[\neg\text{abort}] \leq \Pr[\text{abort}] \\ &\implies 0 \leq \Pr[b = b'] - \frac{1}{2}(1 - \Pr[\text{abort}]) \leq \Pr[\text{abort}] \\ &\implies \left|\Pr[b = b'] - \frac{1}{2}\right| \leq \frac{1}{2}\Pr[\text{abort}] \\ &\implies Adv_{\mathcal{A}}^{(P,f)-DBDH} = \left|\Pr[b = b'] - \frac{1}{2}\right| \leq \frac{(q_{1,\times} + q_{2,\times} + q_e + s + 2)^2 d}{2p} \end{aligned}$$

□

Our schemes are based on a dynamic version of the above (P, f)-DBDH problem. To describe and show the hardness of the problem, we begin with the following lemma.

Lemma 5.1. *Let $P = (p_1, \ldots, p_s), Q = (q_1, \ldots, q_s) \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be two s-tuple of n-variate polynomials over $\mathbb{F}_p$, $f \in \mathbb{F}_p[X_1, \ldots, X_n]$, $O = (P, Q) = (p_1, \ldots, p_s, q_1, \ldots, q_s)$ be a $2s$-tuple of n-variate polynomial. Let T be a variate, $R = (P, QT) = (p_1, \ldots, p_s, q_1T, \ldots, q_sT) = (r_1, \ldots, r_{2s})$ be a $2s$-tuple of $(n+1)$-variate polynomial. If f is not dependent on O, f is not dependent on R.*

Proof. We prove f is not dependent on R by showing $\mathcal{R} = cf^2 + \sum_{k=1}^{2s} b_k r_k f + \sum_{i=1}^{2s}\sum_{j=1}^{2s} a_{i,j} r_i r_j \neq 0$ for any constants $a_{i,j}$, b_k and c that at least one of b_k or c is

non-zero. From the construction of R, we have

$$\begin{aligned}
\mathcal{R} &= cf^2 + \sum_{k=1}^{2s} b_k r_k f + \sum_{i=1}^{2s}\sum_{j=1}^{2s} a_{i,j} r_i r_j \\
&= cf^2 + \sum_{k=1}^{s} b_k p_k f + \sum_{k=1}^{s} b_{s+k} q_k T f \\
&\quad + \sum_{i=1}^{s}\sum_{j=1}^{s} a_{i,j} p_i p_j + \sum_{i=1}^{s}\sum_{j=1}^{s} a_{i+s,j} p_i q_j T \\
&\quad + \sum_{i=1}^{s}\sum_{j=1}^{s} a_{i,j+s} q_i p_j T + \sum_{i=1}^{s}\sum_{j=1}^{s} a_{i+s,j+s} q_i q_j T^2 \\
&= cf^2 + \sum_{k=1}^{s} b_k p_k f + \sum_{i=1}^{s}\sum_{j=1}^{s} a_{i,j} p_i p_j \\
&\quad + T \cdot \left(\sum_{k=1}^{s} b_{s+k} q_k f + \sum_{i=1}^{s}\sum_{j=1}^{s} (a_{i,j+s} + a_{j+s,i}) p_i q_j + \sum_{i=1}^{s}\sum_{j=1}^{s} a_{i+s,j+s} q_i q_j T \right)
\end{aligned}$$

Let $\mathcal{P} = cf^2 + \sum_{k=1}^{s} b_k p_k f + \sum_{i=1}^{s}\sum_{j=1}^{s} a_{i,j} p_i p_j$. Since f is not dependent on $O = (P, Q)$, f is not dependent on P, that is, $\mathcal{P} \neq 0$ where at least one of c or $b_1, \ldots, b_s$ is non-zero. In this case, if $\mathcal{R} = 0$, we have the non-zero polynomial $\mathcal{P}$ involving an extra variate T, which is not possible. Hence, by contradiction, $\mathcal{R} \neq 0$. In the rest case, where all c, $b_1, \ldots, b_s$ are zero, setting $\mathcal{R} = 0$ results in

$$\mathcal{O} = \sum_{k=1}^{s} b_{s+k} q_k f + \sum_{i=1}^{s}\sum_{j=1}^{s} (a_{i,j+s} + a_{j+s,i}) p_i q_j + \sum_{i=1}^{s}\sum_{j=1}^{s} a_{i+s,j+s} q_i q_j T = 0,$$

where at least one of $b_{s+1}, \ldots, b_{2s}$ is non-zero. As f is not dependent on O, we have $\sum_{k=1}^{s} b_{s+k} q_k f + \sum_{i=1}^{s}\sum_{j=1}^{s} (a_{i,j+s} + a_{j+s,i}) p_i q_j \neq 0$. By observing the polynomial $\mathcal{O}$, there is a contradiction that the non-zero polynomial

$$\sum_{k=1}^{s} b_{s+k} q_k f + \sum_{i=1}^{s}\sum_{j=1}^{s} (a_{i,j+s} + a_{j+s,i}) p_i q_j = -\sum_{i=1}^{s}\sum_{j=1}^{s} a_{i+s,j+s} q_i q_j T$$

involves an extra variate T. Hence, by contradiction, $\mathcal{R} \neq 0$ in all cases. Thus f is not dependent on R by definition. □

Lemma 5.2. *Let $P = (p_1, \ldots, p_s), Q = (q_1, \ldots, q_s) \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be two s-tuple of n-variate polynomials over $\mathbb{F}_p$, $f \in \mathbb{F}_p[X_1, \ldots, X_n]$, $O = (P, Q) = (p_1, \ldots, p_s, q_1, \ldots, q_s)$ be a $2s$-tuple of n-variate polynomial. Let $T_1, \ldots, T_\ell$ be ℓ variates, $R = (P, QT_1, \ldots, QT_\ell) = (p_1, \ldots, p_s, q_1T_1, \ldots, q_sT_1, \ldots, q_1T_\ell, \ldots, q_sT_\ell)$ be an $(\ell+1)s$-tuple of $(n+\ell)$-variate polynomial. If f is not dependent on O, f is not dependent on R.*

Proof. Let $R_0 = (P, \underbrace{Q, \ldots, Q}_{\ell})$, $R_1 = (P, QT_1, \underbrace{Q, \ldots, Q}_{\ell-1})$, $\ldots$, $R_\ell = (P, QT_1, \ldots, QT_\ell) = R$. We prove f is not dependent on R by induction.

For the basis $n = 0$, $R_n = R_0$ is equivalent to O, and thus f is not dependent on R_0.

For the inductive step, it is obvious to obtain the result that f is not dependent on R_{n+1} if f is not dependent on R_n by Lemma 5.1.

Hence, by induction, f is not dependent on $R = R_\ell$. □

Definition 5.3 (Decisional ℓ-Combined Bilinear Diffie-Hellman problem)**.** *Let $a, b, c, d, e, f_1, \ldots, f_\ell$ be uniformly and independently chosen from $\mathbb{Z}_p$, $h = g^e$, and Z be uniformly and independently chosen from $\mathbb{G}_1$. Giving two probability distributions $\mathcal{D}_{DCBDH} = (g, g^a, g^b, h, h^c, h^d, \{(g^{f_i}, g^{af_i}, h^{f_i}, h^{af_i})\}_{i=1\ldots\ell}, g^{ab}h^{cd})$ and $\mathcal{D}_{rand} = (g, g^a, g^b, h, h^c, h^d, \{(g^{f_i}, g^{af_i}, h^{f_i}, h^{af_i})\}_{i=1\ldots\ell}, Z)$, there is an algorithm $\mathcal{A}$ can distinguish $\mathcal{D}_{\mathrm{DCBDH}}$ and $\mathcal{D}_{\mathrm{rand}}$ with advantage:*

$$\mathsf{Adv}_{\mathcal{A}}^{\ell-\mathrm{DCBDH}} = \frac{1}{2}\left|\Pr\left[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_{\mathrm{DCBDH}})\right] - \Pr\left[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_{\mathrm{rand}})\right]\right|$$

Alternatively, the problem can be represented as

$$b \in_R \{0,1\}, \quad Z_b = g^{ab}h^{cd}, \quad Z_{1-b} \in_R \mathbb{G}_1,$$
$$\mathsf{Adv}_{\mathcal{A}}^{\ell-\mathrm{DCBDH}} = \left|\Pr\left[b = b' \leftarrow \mathcal{A}\left(\begin{array}{c} g, g^a, g^b, h, h^c, h^d, \\ \{(g^{f_i}, g^{af_i}, h^{f_i}, h^{af_i})\}_{i=1\ldots\ell}, Z_0, Z_1 \end{array}\right)\right] - \frac{1}{2}\right|$$

The ℓ-DCBDH problem belongs to the (P, f)-DBDH problem family. Then we prove that the ℓ-DCBDH problem is hard by showing the advantage $\mathsf{Adv}_{\mathcal{A}}^{\ell-\mathrm{DCBDH}}$ is negligible, using the bilinear group model.

Theorem 5.2. *The lower bound of the advantage $\mathsf{Adv}_{\mathcal{A}}^{\ell-\mathrm{DCBDH}}$ of solving the ℓ-DCBDH problem (Definition 5.3) for the adversary $\mathcal{A}$ is stated as follows with at most q queries to group operations and bilinear pairing operations.*

$$\mathsf{Adv}_{\mathcal{A}}^{\ell-\mathrm{DCBDH}} \leq \frac{3 \cdot (q + 4\ell + 8)^2}{p}$$

Proof. Since the ℓ-DCBDH problem belongs to the (P, f)-DBDH problem family, we rewrite the problem into the problem family by setting

$$P = (1, a, b, ce, de, e, \{(f_i, af_i, ef_i, aef_i)\}_{i=1\ldots\ell}),$$
$$f = ab + cde.$$

Before proving f is not dependent on P, we first prove that f is not dependent on

$$Q = (q_1, \ldots, q_9) = (1, a, b, ce, de, e, f_0, af_0, ef_0, aef_0)$$

where f_0 independently chosen from $\mathbb{Z}_p$. Then we prove f is not dependent on P by Lemma 5.2. Finally, we finish the proof by Theorem 5.1.

To prove f is not dependent on Q, we assume f is dependent on Q that by definition there exists 111 constants $a_{i,j}$, b_k and c such that

$$\mathcal{Q} = cf^2 + \sum_{k=1}^{10} b_k q_k f + \sum_{i=1}^{10}\sum_{j=1}^{10} a_{i,j} q_i q_j = 0$$

where at least one of b_k or c is non-zero. Furthermore, it falls into two cases.

1. $c \neq 0$: In this case, $\mathcal{Q}$ has a term $f^2 = (ab + cde)^2 = a^2b^2 + 2abcde + c^2d^2e^2$. Notably, the term $c^2d^2e^2$ is not involved in $q_k f$ or $q_i q_j$ for any conbination so that f^2 cannot be canceled to make $\mathcal{Q} = 0$. Hence, $\mathcal{Q} \neq 0$ if $c \neq 0$.

2. $c = 0$: In this case, $\mathcal{Q} = \sum_{k=1}^{10} b_k q_k f + \sum_{i=1}^{10}\sum_{j=1}^{10} a_{i,j} q_i q_j$ where at least one b_k is non-zero. Thus $\mathcal{Q}$ has at least one term $q_k f = q_k(ab + cde) = q_k ab + q_k cde$. Stepping in, we focus on the term $q_k cde$. To cancel out the term $q_k cde$ to make $\mathcal{Q} = 0$, we have to construct $q_k cde$ from $q_{k'} f$ or $q_i q_j$ where $k \neq k'$.

   (a) To construct from $q_{k'} f$, we have $q_{k'} f = q_{k'}(ab + cde) = q_{k'} ab + q_{k'} cde$. There are two possible cases:

      i. The first case is $q_{k'} ab = q_k cde$, and thus $q_{k'} = cde$ and $q_k = ab$. However, there is no such a pair found in Q.

      ii. The second case is $q'_k cde = q_k cde$. This case is impossible since $k \neq k'$.

   Thus it is impossible to construct $q_k cde$ from $q_{k'} f$.

   (b) To construct from $q_i q_j$, we analyse the elements in $q_k cde$. Since $q_k cde = q_i q_j$, the variate c is involved in q_k, q_i or q_j. By observing Q, the only polynomial in Q containing c is $q_4 = ce$. Thus $q_k cde = q_i q_4$ and further implies $q_k d = q_i$. Similarly, since $q_k d = q_i$, the variate d is involved in q_k or q_i. Since the only polynomial in Q containing d is $q_5 = de$, we have $q_k d = q_5$ and further $q_k = e = q_6$.

   Hence, the only constructible term $q_k cde$ is cde^2 where $q_k = q_6 = e$. Based on the above result, the only possible constructible $q_k f$ is $q_6 f = e(ab + cde) = abe + cde^2$. To cancel out the term $q_6 f$ to make $\mathcal{Q} = 0$, we have to construct abe from $q_{k'} f$ or $q_i q_j$ where $k' \neq 6$, that is, $q_{k'} \neq e$.

(a) To construct from $q_{k'}f$, we have $q_{k'}f = q_{k'}(ab + cde) = q_{k'}ab + q_{k'}cde$. There are two possible cases:

i. The first case is $q_{k'}ab = abe$ and is impossible since $q_{k'} \neq e$.

ii. The second case is $q'_k cde = abe$, and thus $q_{k'} = \frac{ab}{cd}$. However, there is no such element found in Q.

Thus it is impossible to construct abe from $q_{k'}f$.

(b) To construct from $q_i q_j$, we analyse the elements in abe. Since $abe = q_i q_j$, the variate b is involved in q_i or q_j. By observing Q, the only polynomial in Q containing b is $q_3 = b$. Thus $abe = q_i q_3$ and further implies $q_i = ae$. However, there is no such element ae found in Q.

Hence, it is impossible to derive any $q_k f$ from $q_{k'}f$ or $q_i q_j$ where $k \neq k'$ to make $\mathcal{Q} = 0$. In conclusion, $\mathcal{Q} \neq 0$ if $c = 0$.

Combining all the cases above, $\mathcal{Q} \neq 0$ and it violates the assumption we made. Therefore, by contradiction, f is not dependent on Q.

In the step 2 of this proof, let $P' = (1, a, b, ce, de, e)$, $Q' = (f_0, af_0, ef_0, aef_0)$, and note that $Q = (P', Q')$. Since f is not dependent on Q, by Lemma 5.2, f is not dependent on $R = (P, QT_1, \ldots, QT_\ell) = (1, a, b, ce, de, e, \{(f_0T_i, af_0T_i, ef_0T_i, aef_0T_i)\}_{i=1\ldots\ell})$. As f_0 and T_i are chosen uniformly and independently, f_0T_i is equivalent to f_ℓ. Therefore, P is equivalent to R so that f is not dependent on P.

Finally, we calculate the lower bound of the advantage $\mathsf{Adv}_{\mathcal{A}}^{\ell-\mathrm{DCBDH}}$ by Theorem 5.1 since f is not dependent on P. By observing P and f, we have $s = 6 + 4\ell$, $d_P = 3$, $d_f = 3$, $d = 2 \cdot \max(d_P, d_f) = 6$, and the advantage

$$\mathsf{Adv}_{\mathcal{A}}^{\ell-\mathrm{DCBDH}} \leq \frac{(q+s+2)^2 d}{2p} = \frac{3 \cdot (q+4\ell+8)^2}{p}$$

□

By Theorem 5.2, the ℓ-DCBDH problem is hard as long as ℓ is in polynomial size.

5.3 Formal Definitions

In general, a searchable encryption scheme involves three roles and consists of two encryption parts. In details, the roles are **contributor**, **server** and **user**, and the encryption parts are the message encryption part and the keyword encryption part. A general purpose searchable encryption scheme works as follows. Alice, as a **contributor**, encrypts a file using the message encryption scheme and the

keywords related to that file using the keyword encryption part for the target **users**, including Bob. Let **header** denote the keyword ciphertext, and **payload** denote the file ciphertext. Since a file may be associated with multiple keywords, Alice may generate multiple headers for the payload. After that, Alice assembles the headers and the payload as a single ciphertext, and sends the ciphertext to the **server**. Bob, as one of the target **user**, can ask the **server** to search the ciphertext with certain keywords. To carry out secured search, Bob generates a **trapdoor** for each keyword to be searched, and then uploads the trapdoors to the server via a secure communication channel. Once the server receives the query with the trapdoors from Bob, the server begins to test whether the keywords in the headers match those in the trapdoors. Note that the keywords are not visible to the server, and the headers and trapdoors match only when the corresponding keywords are the same and Bob is one of the intended users that the headers are encrypted for. After searching for all related ciphertexts, the server allows Bob to download the matching payloads. Finally, Bob can download the payloads with matching headers. In addition, a **trusted authority** is required in the identity or attribute-based setting.

5.3.1 Syntax

Formally, we define Linear Encryption with Keyword Search (LEKS) as follows, focusing on the keyword encryption part in a general searchable encryption scheme.

Definition 5.4 (Linear Encryption with Keyword Search)**.** *A Linear Encryption with Keyword Search (LEKS) scheme, involving the contributors, the servers, the users and the trusted authority, consists of the following five probabilistic polynomial time algorithms:*

- $(\mathsf{MSK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(1^\lambda)$*: The randomised system setup algorithm is run by the trusted authority. It takes a security parameter* 1^λ*, and outputs a pair of master secret key* MSK *and public key* PK *for the trusted authority. In the public key scenario where users are identified using public keys generated by themselves, the trusted authority is not required. Thus this algorithm is run by individual users, and outputs a pair of secret key* SK *and public key* PK *for the user.*

- $\mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, I_S)$*: The randomised user key generation algorithm is run by the trusted authority. It takes a master secret key* MSK *and a user identity* I_S*, and generates a user secret key* SK *for the user associated with that identity. In the public key scenario where users are identified using public keys generated by themselves, this algorithm is not used.*

- $C \leftarrow \mathsf{LEKS}(\mathsf{PK}, I_C, W)$: *The randomised keyword encryption algorithm is run by the contributor. It takes a public key* PK*, a target identity* I_C *and a keyword* W*, and outputs a ciphertext* C *of the keyword* W*. To maximum the generality, the target identity* I_C *is viewed as a set that the user* I_S *can access the ciphertext only if* I_S *is in the target identity* I_C*. For instance, the notation* $I_S \in I_C$ *is equivalent to* $P(I_S, I_C) = 1$ *in Predicate Encryption (PE) with the predicate function* P*.*

- $T \leftarrow \mathsf{Trapdoor}(\mathsf{SK}, W)$: *The randomised trapdoor generation algorithm is run by the user. It takes a secret key* SK *and a keyword* W*, and generates a trapdoor* T *of the keyword* W*.*

- $1/0 \leftarrow \mathsf{Test}(C, T)$: *The deterministic test algorithm is run by the server. It takes a ciphertext* $C \leftarrow \mathsf{LEKS}(\mathsf{PK}, I_C, W)$ *and a trapdoor* $T \leftarrow \mathsf{Trapdoor}(\mathsf{SK}, W)$ *where* $\mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, I_S)$*, and outputs*

$$\begin{cases} 1 & \text{if } W = W' \wedge I_S \in I_C, \\ 0 & \text{otherwise.} \end{cases}$$

In addition, the scheme is required to be correct.

Definition 5.5 (Correctness)**.** *A LEKS scheme is correct if the following statement is always true:*

$$\forall(\mathsf{MSK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(1^\lambda), \quad \forall I_C, \quad \forall I_S \in I_C, \quad \forall \mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, I_S),$$
$$\forall W \in \{0,1\}^*, \quad \forall C \leftarrow \mathsf{LEKS}(\mathsf{PK}, I_C, W), \quad \forall T \leftarrow \mathsf{Trapdoor}(\mathsf{SK}, W),$$
$$\mathsf{Test}(C, T) = 1.$$

5.3.2 Security Model

In LEKS, we consider that the server is *honest but curious*. In addition, we do not consider the Keyword Guessing Attack (KGA), since the server can always generate ciphertexts with certain keywords to test with the trapdoor legitimately. However, we can prevent anyone from extracting the keyword directly from the trapdoor by applying an one-way function such as a preimage-resistant hash function.

We present security under two versions of attacks: Indistinguishability under Selective-ID Adaptive Chosen Keyword Attacks (IND-sCKA) and Indistinguishability under Adaptive Chosen Keyword Attacks (IND-CKA). We first define the IND-sCKA game (Fig. 5.1) where an adaptive adversary $\mathcal{A}$ tries to distinguish a ciphertext generated from keywords either W_0 or W_1:

1. The adversary selects a target identity set I_T and submits it to the challenger.

2. The challenger runs the $\mathsf{Setup}(1^\lambda)$ to generate a pair of master secret key MSK and public key PK. Then the challenger passes the public key PK to the adversary $\mathcal{A}$.

3. The adversary can adaptively ask the challenger the secret key SK of the user with identity I by querying the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}$. At the same point, the challenger records the queried identity I in the identity list $\mathcal{I}$. The restriction is that the queried identity I must not be in the target identity set I_T.

4. The adversary can adaptively ask the challenger the trapdoor T of the user identity I with the keyword W by querying the trapdoor generation oracle $\mathcal{O}_{\mathsf{Trapdoor}}$. If the queried identity I is not in the target identity set I_T, it can be resolved that the adversary queries the oracle $\mathcal{O}_{\mathsf{KeyGen}}$ to obtain the secret key SK of the identity I. After that, the adversary obtains the trapdoor T by running the algorithm $\mathsf{Trapdoor}$ using the secret key SK. Otherwise, the challenger runs the algorithm KeyGen and then the algorithm $\mathsf{Trapdoor}$ to get the trapdoor, and passes it to the adversary. At the same point, the challenger records the queried keyword W in the keyword list $\mathcal{W}$.

5. At some point, the adversary $\mathcal{A}$ outputs two keywords W_0 and W_1 to be challenged where those two keywords must not be in the keyword list $\mathcal{W}$.

6. The challenger randomly selects b to be either 0 or 1 uniformly. Then it generates a ciphertext $C \leftarrow \mathsf{LEKS}(\mathsf{PK}, I_T, W_b)$ and passes it to the adversary.

7. The adversary can continue to query all oracles with the same restriction. Note that the adversary cannot query the target keywords W_0 and W_1 to the oracle $\mathcal{O}_{\mathsf{Trapdoor}}$.

8. Eventually, the adversary outputs a bit b'. The adversary wins the game if $b = b'$.

We define the advantage of winning the IND-sCKA game (Fig. 5.1) as follows

$$\mathsf{Adv}_{\mathcal{A}}^{\text{IND-sCKA}} = \left| \Pr\left[b = b' \wedge \mathcal{I} \cap I_T = \emptyset \wedge W_0, W_1 \notin \mathcal{W} \right] - \frac{1}{2} \right|$$

Definition 5.6 (IND-sCKA Security)**.** *A LEKS scheme is Indistinguishable under Selective-ID Adaptive Chosen Keyword Attack (IND-sCKA) if* $\mathsf{Adv}_{\mathcal{A}}^{\text{IND-sCKA}}$ *is a negligible function for all adversary* $\mathcal{A}$ *winning the IND-sCKA game (Fig. 5.1) in polynomial time.*

$\underline{\mathsf{Game}^{\lambda}_{\text{IND-sCKA}}}$

$\mathcal{I}, \mathcal{W} \leftarrow \emptyset$

$I_T \leftarrow \mathcal{A}$

$(\mathsf{MSK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(1^{\lambda})$

$(W_0, W_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}}, \mathcal{O}_{\mathsf{Trapdoor}}}(\mathsf{PK})$

$b \in_R \{0, 1\}$

$C \leftarrow \mathsf{LEKS}(\mathsf{PK}, I_T, W_b)$

$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}}, \mathcal{O}_{\mathsf{Trapdoor}}}(C)$

$\underline{\mathcal{O}_{\mathsf{KeyGen}}}$

$\mathcal{I} \leftarrow \mathcal{I} \cup \{I\}$

return $\mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, I)$

$\underline{\mathcal{O}_{\mathsf{Trapdoor}}}$

$\mathcal{W} \leftarrow \mathcal{W} \cup \{W\}$

return $T \leftarrow \mathsf{Trapdoor}(\mathsf{SK}, W)$

$$\mathsf{Adv}^{\text{IND-sCKA}}_{\mathcal{A}} = \left| \Pr\left[b = b' \wedge \mathcal{I} \cap I_T = \emptyset \wedge W_0, W_1 \notin \mathcal{W}\right] - \frac{1}{2} \right|$$

Figure 5.1: IND-sCKA Game

$\underline{\mathsf{Game}^{\lambda}_{\text{IND-CKA}}}$

$\mathcal{I}, \mathcal{W} \leftarrow \emptyset$

$(\mathsf{MSK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(1^{\lambda})$

$(I_T, W_0, W_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}}, \mathcal{O}_{\mathsf{Trapdoor}}}(\mathsf{PK})$

$b \in_R \{0, 1\}$

$C \leftarrow \mathsf{LEKS}(\mathsf{PK}, I_T, W_b)$

$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}}, \mathcal{O}_{\mathsf{Trapdoor}}}(C)$

$\underline{\mathcal{O}_{\mathsf{KeyGen}}}$

$\mathcal{I} \leftarrow \mathcal{I} \cup \{I\}$

return $\mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, I)$

$\underline{\mathcal{O}_{\mathsf{Trapdoor}}}$

$\mathcal{W} \leftarrow \mathcal{W} \cup \{W\}$

return $T \leftarrow \mathsf{Trapdoor}(\mathsf{SK}, W)$

$$\mathsf{Adv}^{\text{IND-CKA}}_{\mathcal{A}} = \left| \Pr\left[b = b' \wedge \mathcal{I} \cap I_T = \emptyset \wedge W_0, W_1 \notin \mathcal{W}\right] - \frac{1}{2} \right|$$

Figure 5.2: IND-CKA Game

Next, we define the IND-CKA game (Fig. 5.2), which is similar as the IND-sCKA game Fig. 5.1. The difference between IND-sCKA and IND-CKA is that the adversary is given the public key PK in IND-CKA before submitting the target identity set I_T. Similarly, we define the IND-CKA security for LEKS schemes.

Definition 5.7 (IND-CKA Security)**.** *A LEKS scheme is Indistinguishable under Adaptive Chosen Keyword Attack (IND-CKA) if* $\mathsf{Adv}^{\text{IND-CKA}}_{\mathcal{A}}$ *is a negligible function for all adversary* $\mathcal{A}$ *winning the IND-CKA game (Fig. 5.2) in polynomial time.*

5.4 Linear Encryption Template

In this section, we define the Linear Encryption Template (LET). Informally, a LET models an asymmetric encryption scheme, consisting of the **senders**, the **recipients** and the **trusted authority**. In LET, Alice, as the **recipient**, gets her secret key from the **trusted authority** using her identity where her identity is her public key. If LET is modelling a PKE scheme, Alice's secret/public key pair is generated by herself. Thus the **trusted authority** is not required. To securely send a message

to a set of recipients, including Alice, Bob, as the **sender**, encrypts the message into a ciphertext, and sends it to Alice. Once Alice receives the ciphertext, she can decrypts and obtains the original message if and only if she is one of the target recipients. Furthermore, if an encryption scheme fits in the LET, we can use it to construct the corresponding LEKS in Section 5.5. Formally, we describe the definition of LET as follows.

Definition 5.8 (Linear Encryption Template)**.** *A Linear Encryption Template (LET), involving the senders, the recipients, and the trusted authority, consists of the following four probabilistic polynomial algorithms:*

- $(\mathsf{MSK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(\mathsf{param}, \alpha)$*: The randomised system setup is run by the trusted authority. It takes a set of system parameters, such as the description of groups, security parameters and randomnesses, and it reuses these parameters. The algorithm also takes a component α, which is used to create the ciphertext. The output of this algorithm is a pair of master secret key* MSK *and public key* PK *of the trusted authority. If there is no trusted authority that users generate their key pairs by themselves, this algorithm is run by the user, and outputs a pair of user secret key* SK *and public key* SK*.*

- $\mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, I_S)$*: The randomised user key generation algorithm is run by the trusted authority. It takes a master secret key* MSK *and a user identity I_S, and generates a user secret key* SK *for the user associated with that identity. If there is no trusted authority that users generate their key pairs by themselves, this algorithm is not used.*

- $C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, I_C, M, s)$*: The randomised encryption algorithm is run by the sender. It takes a public key* PK*, a target identity set I_C, a message M and a randomness s, and outputs a ciphertext C of the message M. The randomness s is used to bind the ciphertext parts in C and further to bind other ciphertext parts when constructing LEKS schemes. It is required that the ciphertext must be in the form of $C = (C_0, C_1, \dots)$ where $C_0 = M \cdot e(g,g)^{\alpha s}$.*

- $M \leftarrow \mathsf{Decrypt}(SK, C)$*: The deterministic decryption algorithm is run by the recipient. It takes a secret key* SK *and a ciphertext C, and outputs the original message M. The decryption process is required to be two steps. The first step is to run the sub-decryption algorithm $\mathcal{D}$ to get $e(g,g)^{\alpha s} \leftarrow \mathcal{D}(\mathsf{SK}, C_1, \dots)$. Then the second step is to extract the message $M = \frac{C_0}{e(g,g)^{\alpha s}}$. Importantly, the sub algorithm $\mathcal{D}$ is required to have* ***linearity****:*

$$\forall t \in \mathbb{Z}_p, \quad \mathcal{D}(\mathsf{SK}^t, C_1, \dots) = D(\mathsf{SK}, C_1, \dots)^t$$

If SK *consists multiple elements that* $\mathsf{SK} = (\mathsf{SK}_1, \mathsf{SK}_2, \dots)$*, the term* SK^t *denotes* $(\mathsf{SK}_1^t, \mathsf{SK}_2^t, \dots)$*.*

In addition, the scheme is required to be correct.

Definition 5.9 (Correctness)**.** *A LET scheme is correct if the following statement is always true:*

$$\forall(\mathsf{MSK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(\mathsf{param}, \alpha), \quad \forall I_C, \quad \forall I_S \in I_C, \quad \forall \mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, I_S),$$
$$\forall M \in \mathbb{G}_2, \quad \forall s \in \mathbb{Z}_p, \quad \forall C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, I_C, M, s), \quad \mathsf{Decrypt}(\mathsf{SK}, C) = M.$$

5.5 Generic Construction from Linear Encryption Template

In this section, we build our LEKS scheme with from a LET scheme as the keyword encryption part. To construct a fully searchable encryption scheme, we can reuse the LET scheme as the message encryption part, and combine with the LEKS. Alternatively, we also can use other encryption schemes as the message encryption part. The main idea of the construction is to use the LET part for authentication and combine it with a keyword equality test with the same randomness.

Let $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a LET modelled encryption scheme. Our LEKS scheme works as follows.

– $(\mathsf{MSK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(1^\lambda)$: Given a security parameter 1^λ, the algorithm generates two groups $\mathbb{G}_1$, $\mathbb{G}_2$ of prime order p, and specifies a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. the algorithm also selects a random generator g of the group $\mathbb{G}_1$, and a preimage resistant hash function $H : \{0,1\}^* \to \mathbb{G}_1$, which may be modelled as an random oracle. After that, the algorithm randomly chooses two randomness $x_1, x_2 \in_R \mathbb{Z}_p$, and calculates $g_1 = g^{x_1}$ and $g_2 = g^{x_2}$. Then the algorithm packs all above elements into param, sets $\alpha = x_1 x_2$, and passes to the algorithm $\Pi.\mathsf{Setup}$ to obtain the key pair $\Pi.\mathsf{MSK}$ and $\Pi.\mathsf{PK}$. Finally, the algorithm keeps the master secret key $\mathsf{MSK} = \Pi.\mathsf{MSK}$, and publishes the public key $\mathsf{PK} = (\mathbb{G}_1, \mathbb{G}_2, e, g, g_1, g_2, \Pi.\mathsf{PK})$.

$$\mathbb{G}_1 = \langle g \rangle, \quad e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2, \quad H : \{0,1\}^* \to \mathbb{G}_1,$$
$$x_1, x_2 \in_R \mathbb{Z}_p^+, \quad g_1 = g^{x_1}, \quad g_2 = g^{x_2},$$
$$\mathsf{param} = (\mathbb{G}_1, \mathbb{G}_2, e, g, H, x_1, x_2),$$
$$(\Pi.\mathsf{MSK}, \Pi.\mathsf{PK}) \leftarrow \Pi.\mathsf{Setup}(\mathsf{param}, x_1 x_2)$$

return $(\mathsf{MSK}, \mathsf{PK}) = (\Pi.\mathsf{MSK}, (\mathbb{G}_1, \mathbb{G}_2, e, g, g_1, g_2, H, \Pi.\mathsf{PK}))$.

– $\mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, I_S)$: For key generation, the algorithm $\Pi.\mathsf{KeyGen}$ is directly invoked. Note that $\mathsf{MSK} = \Pi.\mathsf{MSK}$.
return $\mathsf{SK} \leftarrow \Pi.\mathsf{KeyGen}(\mathsf{MSK}, I_S)$.

– $C \leftarrow \mathsf{LEKS}(\mathsf{PK}, I_C, W)$: To encrypt a keyword W for a target identity set I_C, the algorithm randomly chooses two randomness $r_1, r_2 \in_R \mathbb{Z}_p$. Then it computes $C'_1 = g_2^{r_2} H(W)^{r_1}$ and $C'_2 = g_1^{r_1}$ to encrypt the keyword W. After that, the algorithm invokes $\Pi.\mathsf{Encrypt}$ with the randomness r_2 to get the ciphertext $(C_0, C_1, \dots)$ to assure the target identity set I_C. Finally, the algorithm assembles two parts together $C = (C'_1, C'_2, C_1, \dots)$ as the full ciphertext bound using r_2 where $C_0 = M \cdot e(g,g)^{x_1 x_2 r_2}$ is dropped. Since C_0 is not used in C, we can safely setting the message M to 0 when invoking $\Pi.\mathsf{Encrypt}$.

$$r_1, r_2 \in_R \mathbb{Z}_p^+, C'_1 = g_2^{r_2} H(W)^{r_1}, C'_2 = g_1^{r_1}$$
$$(C_0, C_1, \dots) \leftarrow \Pi.\mathsf{Encrypt}(\mathsf{PK}, I_C, 0, r_2)$$

return $C = (C'_1, C'_2, C_1, \dots)$.

– $T \leftarrow \mathsf{Trapdoor}(\mathsf{SK}, W)$: To generate a trapdoor of the keyword W, the algorithm randomly selects a randomness $s \in_R \mathbb{Z}_p^+$. Then it calculates $T = (T_1, T_2, T_3)$ where $T_1 = g_1^s$, $T_2 = H(W)^s$ and $T_3 = \mathsf{SK}^s$. For SK^s, the operation works the same as in Definition 5.8.

$$s \in_R \mathbb{Z}_p^+, T_1 = g_1^s, T_2 = H(W)^s, T_3 = \mathsf{SK}^s$$

return $T = (T_1, T_2, T_3)$.

– $1/0 \leftarrow \mathsf{Test}(C, T)$: For equality test of both the keyword and the identity, the algorithm tests the equality of the following return statement.
return $e(C'_1, T_1)/e(C'_2, T_2) \stackrel{?}{=} \Pi.\mathcal{D}(T_3, C_1, \dots)$.

We verify that our LEKS conversion is correct.

Theorem 5.3. *The proposed conversion from the LET scheme to the LEKS scheme is correct if the corresponding encryption scheme modelled by the LET scheme is correct.*

Proof. The correctness is verified as follows. First, we calculate the left hand side

of the test equation.

$$\begin{aligned} E_1 &= e(C'_1, T_1)/e(C'_2, T_2) \\ &= \frac{e(g_2^{r_2} H(W)^{r_1}, g_1^s)}{e(g_1^{r_1}, H(W)^s)} \\ &= \frac{e(g_2^{r_2}, g_1^s) \cdot e(H(W)^{r_1}, g_1^s)}{e(g_1^{r_1}, H(W)^s)} \\ &= e(g_1, g_2)^{r_2 s} \end{aligned}$$

Then we calculate the right hand side of the test equation.

$$\begin{aligned} E_2 &= \Pi.\mathcal{D}(T_3, C_1, \dots) \\ &= \Pi.\mathcal{D}(\mathsf{SK}^s, C_1, \dots) \\ &= \Pi.\mathcal{D}(\mathsf{SK}, C_1, \dots)^s \\ &= e(g, g)^{x_1 x_2 r_2 s} \\ &= e(g_1, g_2)^{r_2 s} \end{aligned}$$

As $E_1 = E_2$, the correctness is proved. □

However, we are uncertain for the security of the above construction, since some components are shared outside the encryption Π that may break the security of Π in its original model. Therefore, we require an individual security proof for each conversion to ensure the security. For this reason, we provide the related security proof for each conversion in the following sections.

5.6 Public-Key Encryption with Keyword Search

In this section, we show a simple instance of our LEKS conversion that we convert a PKE scheme into a PEKS scheme. Af first, we propose a simple variant of ElGamal encryption [ElG85]. Then we convert it into a LEKS scheme by the method in Section 5.5. Finally, we prove the resulted LEKS scheme is IND-CKA secure in random oracle model.

5.6.1 Base Scheme

In this variant of ElGamal Encryption scheme, users generates their key pairs by themselves so that there is no trusted authority. Thus the algorithm KeyGen in LET is not used. The encryption scheme works as follows, modelled by LET.

– $(\mathsf{SK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(\mathsf{param}, \alpha)$: The key generation algorithm reuses the parameters param and sets g^α as the secret key. It also publish the public key

$\mathsf{PK} = e(g,g)^{\alpha}$.
return $(\mathsf{SK}, \mathsf{PK}) = (g^{\alpha}, e(g,g)^{\alpha})$.

– $C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, M, t)$: Since there is no second level of the identity, the target identity set I_C is not used. Instead, the public key PK is used as the target identity. As the same as ElGamal encryption, the ciphertext is computed as $C = (C_0, C_1)$ where $C_0 = M \cdot \mathsf{PK}^t = M \cdot e(g,g)^{\alpha t}$ and $C_1 = g^t$, reusing the randomness t from the upper level.
return $C = (C_0, C_1) = (M \cdot e(g,g)^{\alpha t}, g^t)$.

– $M \leftarrow \mathsf{Decrypt}(\mathsf{SK}, C)$: To decrypt, the algorithm computes $M = \frac{C_0}{e(SK, C_1)}$ where $\mathcal{D}(\mathsf{SK}, C_1) = e(\mathsf{SK}, C_1)$. Notably, the sub-algorithm $\mathcal{D}$ has linearity that

$$\forall s \in \mathbb{Z}_p, \quad \mathcal{D}(\mathsf{SK}^s, C_1) = e(\mathsf{SK}^s, C_1) = e(\mathsf{SK}, C_1)^s = \mathcal{D}(\mathsf{SK}, C_1)^s$$

We show that this variant is correct.

Theorem 5.4 (Correctness)**.** *The above variant of ElGamal Encryption scheme is correct.*

Proof. Since

$$\frac{C_0}{e(\mathsf{SK}, C_1)} = \frac{M \cdot e(g,g)^{\alpha t}}{e(g^{\alpha}, g^t)} = \frac{M \cdot e(g,g)^{\alpha t}}{e(g,g)^{\alpha t}} = M,$$

we have proved the correctness. □

5.6.2 Construction from the Base Scheme

In this subsection, we apply the LEKS conversion as follows with some key notes.

– $(\mathsf{SK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(1^{\lambda})$: The public key of LET is $e(g,g)^{x_1 x_2}$, which can be calculated by $e(g^{x_1}, g^{x_2}) = e(g_1, g_2)$. Thus this element is removed for optimisation.

$$\mathbb{G}_1 = \langle g \rangle, \quad e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2, \quad H : \{0,1\}^* \to \mathbb{G}_1,$$
$$x_1, x_2 \in_R \mathbb{Z}_p, \quad g_1 = g^{x_1}, \quad g_2 = g^{x_2},$$
$$\mathsf{SK} = g^{x_1 x_2}, \quad \mathsf{PK} = (\mathbb{G}_1, \mathbb{G}_2, e, g, g_1, g_2, H)$$

return (SK, PK).

– $C \leftarrow \mathsf{LEKS}(\mathsf{PK}, W)$:

$$r_1, r_2 \in_R \mathbb{Z}_p, \quad C_1 = g_2^{r_2} H(W)^{r_1}, \quad C_2 = g_1^{r_1}, \quad C_3 = g^{r_2}$$

return $C = (C_1, C_2, C_3)$.

– $T \leftarrow \mathsf{Trapdoor}(\mathsf{SK}, W)$:

$$s \in_R \mathbb{Z}_p, \quad T_1 = g_1^s, \quad T_2 = H(W)^s, \quad T_3 = g^{x_1 x_2 s}$$

return $T = (T_1, T_2, T_3)$.

– $1/0 \leftarrow \mathsf{Test}(C, T)$: Original, the converted equation is $e(C_1, T_1)/e(C_2, T_2) \stackrel{?}{=} e(T_3, C_3)$. In order to optimise the computing efficiency, it is rewritten as the following return statement.
return $e(C_1, T_1) \stackrel{?}{=} e(C_2, T_2) \cdot e(C_3, T_3)$.

5.6.3 Security Proof

Theorem 5.5. *The proposed PEKS scheme is IND-CKA (Definition 5.7) secure. If an adversary $\mathcal{A}$ can win the IND-CKA game (Fig. 5.2) with the advantage ε, an algorithm $\mathcal{S}$ can be constructed to solve ℓ-DCBDH problem (Definition 5.3) in polynomial time the advantage $\varepsilon' \geq \frac{\varepsilon}{2e(q+2)}$, querying $\mathcal{O}_{\mathsf{Trapdoor}}$ for at most ℓ times.*

Proof. Let $\delta = (g, g^a, g^b, h, h^c, h^d, \{(g^{f_i}, g^{af_i}, h^{f_i}, h^{af_i})\}_{i=1\ldots\ell}, Z)$ be an ℓ-DCBDH instance (Definition 5.3) to be challenged to the simulator $\mathcal{S}$ for distinguishing that $\delta \in \mathcal{D}_{\text{DCBDH}}$ or $\delta \in \mathcal{D}_{\text{rand}}$. Note that the simulator $\mathcal{S}$ does not make use of all tree elements in the tuple $(g^{f_i}, g^{af_i}, h^{f_i}, h^{af_i})$. Instead, only (h^{f_i}, h^{af_i}) is used. In other words, the proposed PEKS can be reduced to a weaker version of ℓ-DCBDH assumption. In addition, the simulator $\mathcal{S}$ gets two groups $\mathbb{G}_1$, $\mathbb{G}_2$ with a bilinear map e from the instance δ. The simulator $\mathcal{S}$ sets $g_1 = h$, $g_2 = g^a$, and passes $\mathsf{PK} = (\mathbb{G}_1, \mathbb{G}_2, e, g, g_1, g_2)$ to the adversary $\mathcal{A}$. Besides that, the simulator $\mathcal{S}$ simulates the following two oracles.

– $\mathcal{O}_H$: The hash function H is viewed as a random oracle. Upon requesting the hash value of the keyword W_i, the simulator $\mathcal{S}$ randomly tosses a coin $c_i \in_R \{0, 1\}$ such that $\Pr[c_i = 0] = \alpha$ where the value of α is determined later. The simulator $\mathcal{S}$ also randomly chooses $a_i \in_R \mathbb{Z}_p^+$, and computes the hash value

$$h_i = \begin{cases} h^c h^{a_i} & \text{if } c_i = 0 \\ h^{a_i} & \text{if } c_i = 1 \end{cases}$$

Due to the randomness a_i, the distribution of $\{h_i\}$ is indistinguishable with a random distribution of $\mathbb{G}_1$. Finally, the simulator $\mathcal{S}$ returns the hash value h_i to the adversary $\mathcal{A}$. At the same time, the simulator $\mathcal{S}$ maintains hash list $\mathcal{H} = \{W_i, c_i, a_i, h_i\}$. If the requested keyword W_i is already in the list $\mathcal{H}$, the simulator $\mathcal{S}$ returns the corresponding h_i to the adversary $\mathcal{A}$.

– $\mathcal{O}_{\mathsf{Trapdoor}}$: To generate a trapdoor for the keyword W_i, the simulator $\mathcal{S}$ invokes $\mathcal{O}_H$ with the keyword W_i. If the corresponding $c_i = 0$, the simulator $\mathcal{S}$ **aborts**. Otherwise, the simulator $\mathcal{S}$ obtains $h_i = h^{a_i}$ from the list $\mathcal{H}$. Then the simulator $\mathcal{S}$ gets a tuple (h^{f_q}, h^{af_q}) from the instance δ for the q-th query. After that, the simulator $\mathcal{S}$ computes the trapdoor $T = (T_1, T_2, T_3)$. Let f_q be the randomness s used in the algorithm $\mathsf{Trapdoor}$. The simulator $\mathcal{S}$ calculates

$$\begin{aligned} T_1 &= g_1^s = h^{f_q} \\ T_2 &= H(W)^s = (h^{a_i})^{f_q} = (h^{f_q})^{a_i} \\ T_3 &= g^{x_1 x_2 s} = g_1^{x_2 s} = h^{af_q}. \end{aligned}$$

Finally, the simulator $\mathcal{S}$ returns T to the adversary $\mathcal{A}$.

Since the PEKS scheme does not have a KeyGen algorithm, the oracle $\mathcal{O}_{\mathsf{KeyGen}}$ is not provided, and the target identity set I_T is not required.

At some point, the adversary $\mathcal{A}$ outputs two target keywords W_0 and W_1. Then the simulator $\mathcal{S}$ invokes $\mathcal{O}_H$ for the hash value of W_0 and W_1. If the corresponding c_0, c_1 to the keywords W_0 and W_1 is equal to 1, the simulator $\mathcal{S}$ **aborts**. Otherwise, the simulator $\mathcal{S}$ randomly selects $b \in_R \{0, 1\}$ such that $c_b = 0$. In other words, the simulator $\mathcal{S}$ always has $H(W_b) = h^c h^{a_i}$. Before computing the ciphertext $C = (C_1, C_2, C_3)$, we observe the genuine LEKS algorithm. Let (d, b) be the randomnesses (r_1, r_2) used in the algorithm LEKS. We have

$$\begin{aligned} C_1 &= g_2^{r_2} H(W)^{r_1} = g^{ab}(h^c h^{a_i})^d = g^{ab} h^{cd} \cdot (h^d)^{a_i}, \\ C_2 &= g_1^{r_1} = h^d, \\ C_3 &= g^{r_2} = g^b. \end{aligned}$$

The simulator $\mathcal{S}$ replaces $g^{ab}h^{cd}$ with Z in the instance δ:

$$C_1 = Z \cdot (h^d)^{a_i}, \quad C_2 = h^d, \quad C_3 = g^b.$$

The resulted ciphertext C is consistent only if $Z = g^{ab}h^{cd}$. After received the ciphertext C from the simulator $\mathcal{S}$, the adversary $\mathcal{A}$ can continue to query all the oracles as before with defined restrictions in IND-CKA game. Eventually, the adversary $\mathcal{A}$ outputs a bit b'. If $b = b'$, the ciphertext C is believed consistent that $Z = g^{ab}h^{cd}$, and the simulator $\mathcal{S}$ outputs $\delta \in \mathcal{D}_{\mathrm{DCBDH}}$. Otherwise, the simulator $\mathcal{S}$ outputs $\delta \in \mathcal{D}_{\mathrm{rand}}$.

Lemma 5.3. *Let ρ be the probability that the simulator $\mathcal{S}$ does not abort. Assuming the probability of $\delta \in \mathcal{D}_{\mathrm{DCBDH}}$ is $\frac{1}{2}$ and the advantage of the adversary $\mathcal{A}$ winning the IND-CKA game (Fig. 5.2) is ε, the advantage ε' of the simulator $\mathcal{S}$ solving the*

ℓ-DCBDH problem is at least $\frac{\rho\varepsilon}{2}$.

Proof. Let E_s be the event that the simulator $\mathcal{S}$ successfully solves the ℓ-DCBDH problem, E_w be the event that the adversary $\mathcal{A}$ wins in the IND-CKA game (Fig. 5.2) that $\Pr[E_w] = \frac{1}{2} + \varepsilon$, E_a be the event that $\mathcal{S}$ aborts, and E_r be the event that $\mathcal{S}$ solves the problem with random guess that $\Pr[E_r] = \frac{1}{2}$. If $\delta \in \mathcal{D}_{\text{rand}}$, the behaviour of the adversary $\mathcal{A}$ is unpredictable. Therefore, the adversary $\mathcal{A}$ wins the IND-CKA game (Fig. 5.2) better than a random guess that $\Pr[E_w] \geq \Pr[E_r]$. Thus we have the probability that $\mathcal{S}$ successfully solves the ℓ-DCBDH problem.

$$\begin{aligned}
\Pr[E_s] &= \frac{1}{2}\Pr[E_s \mid \delta \in \mathcal{D}_{\text{DCBDH}}] + \frac{1}{2}\Pr[E_s \mid \delta \in \mathcal{D}_{\text{rand}}] \\
&\geq \frac{1}{2}\left(\Pr[E_w]\Pr[\neq E_a] + \Pr[E_r]\Pr[E_a]\right) + \frac{1}{2}\Pr[E_r] \\
&= \frac{1}{2}\left((\frac{1}{2} + \varepsilon)\rho + \frac{1}{2}(1-\rho)\right) + \frac{1}{2}\cdot\frac{1}{2} \\
&= \frac{1}{2}\rho\varepsilon + \frac{1}{2}
\end{aligned}$$

Finally, we calculate the advantage ε' as follows.

$$\varepsilon' = \left|\varepsilon - \frac{1}{2}\right| \geq \frac{1}{2}\rho\varepsilon$$

□

Lemma 5.4. *The probability ρ that the simulator $\mathcal{S}$ does not abort is at least $\frac{1}{e(q+2)}$ with q times of $\mathcal{O}_{\mathsf{Trapdoor}}$ queries.*

Proof. There are two possible points that the simulator $\mathcal{S}$ may abort.

1. The simulator $\mathcal{S}$ aborts in answering $\mathcal{O}_{\mathsf{Trapdoor}}$ if $c_i = 0$. The probability $\Pr[E_1]$ of $\mathcal{S}$ not aborting is $\Pr[E_1] = (1-\alpha)^q$.

2. The simulator $\mathcal{S}$ aborts in the challenge phase if $c_0 = c_1 = 1$. The probability $\Pr[E_2]$ of $\mathcal{S}$ not aborting if $\Pr[E_2] = 1 - (1-\alpha)^2 = \alpha(1-\alpha)$.

Since all events are independent, we have

$$\rho = \Pr[E_1]\cdot\Pr[E_2] = (1-\alpha)^q \cdot \alpha(1-\alpha) = \alpha(1-\alpha)^{q+1}.$$

The value of ρ is maximised when $\alpha = \frac{1}{q+2}$. Therefore, we have

$$\begin{aligned}\rho &= \frac{1}{q+2}\left(1-\frac{1}{q+2}\right)^{q+1} \\ &\geq \frac{1}{q+2}\lim_{q\to\infty}\left(1-\frac{1}{q+2}\right)^{q+1} \\ &= \frac{1}{e(q+2)}\end{aligned}$$

□

Combining the above two lemmas, we have

$$\varepsilon' \geq \frac{1}{2}\rho\varepsilon \geq \frac{\varepsilon}{2e(q+2)}$$

□

5.7 Key-Policy Attribute-Based Keyword Search

In this section, we show a useful instance of our LEKS conversion by converting a Key-Policy Attribute-Based Encryption (KP-ABE) into a Key-Policy Attribute-Based Encryption with Keyword Search (KP-ABKS) scheme. We starts with a KP-ABE scheme [PTMW06] which is a variant of Goyal et al.'s scheme [GPSW06] while the function T defined in [GPSW06] is replaced with a random oracle. Then we convert it into a KP-ABKS scheme by the method in Section 5.5. Finally, we prove the resulted KP-ABKS scheme is IND-sCKA secure in random oracle model.

5.7.1 Base Scheme

The variant scheme [PTMW06] of Goyal et al.'s scheme [GPSW06] modelled by LET works as follows.

- $(\mathsf{MSK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(\mathsf{param}, \alpha)$: The key generation algorithm reuses the parameters param, $g_1 = g^{x_1}$, and $g_2 = g^{x_2}$. Then master secret key is $y = x_1$. Since the function T is replaced with a random oracle, the algorithm is required to choose a cryptographic hash function $H : \{0,1\}^* \to \mathbb{G}_1$.
  return $(\mathsf{MSK}, \mathsf{PK}) = (x_1, (g_1, g_2, H))$.

- $\mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, I_S)$: In KP-ABE, the user identity set I_S is the policy modelled as an access tree $\mathcal{T}$ (details can be found in [GPSW06]). The algorithm chooses a random polynomial q_x for each non-leaf node $x \in \mathcal{T}$ in a top-down manner. For each non-leaf node x, the degree d_x of the polynomial

q_x is $d_x = k_x - 1$ where k_x is the threshold value of that node. For the root node, the algorithm sets $q_{\text{root}}(0) = x_1$. For other nodes, the algorithm sets $q_x(0) = q_{\text{parent}(x)}(\text{index}(x))$. With polynomials for the access tree $\mathcal{T}$ is decided, the algorithm generates the secret key components for the user. For each leaf node x, the algorithm chooses a random number $r_x \in_R \mathbb{Z}_p$, and calculates

$$D_x = g_2^{q_x(0)} H(\mathsf{attr}(x))^{r_x}$$
$$R_x = g^{r_x}$$

return $\mathsf{SK} = (\mathcal{T}, \{(D_x, R_x)\}_{x \in \text{leaves}(\mathcal{T})})$.

– $C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, I_C, M, t)$: In KP-ABE, the target identity set I_C is the attributes γ. To encrypt, the algorithm calculates $C_0 = M \cdot e(g_1, g_2)^t$, $C_1 = g^t$, $C_2 = \gamma$. For each attribute $\mathsf{attr}_i \in \gamma$, the algorithm computes $C_i = H(\mathsf{attr}_i)^t$. As required by LET, we note that $C_0 = M \cdot e(g_1, g_2)^t = e(g^{x_1}, g^{x_2})^t = e(g, g)^{x_1 x_2 t} = e(g, g)^{\alpha t}$. Return $C = (C_0, C_1, C_2, \{C_i\}_{\mathsf{attr}_i \in \gamma})$.

– $M \leftarrow \mathsf{Decrypt}(\mathsf{SK}, C)$: At first, the algorithm checks whether $\mathcal{T}(\gamma) = 1$ or not. If the attributes do not match the policy that $\mathcal{T}(\gamma) = 0$, the algorithms returns $\perp$. Otherwise, the algorithm proceeds the sub-algorithm $\mathcal{D}$ as follows. For those matching attributes $\mathsf{attr}_i = \mathsf{attr}(x)$, where $\mathsf{attr}_i \in \gamma$ and leaf node $x \in \mathcal{T}$, the algorithm can decrypt that node by calculating

$$F_x = \frac{e(D_x, C_1)}{e(R_x, C_i)} = \frac{e(g_2^{q_x(0)} H(\mathsf{attr}(x))^{r_x}, g^t)}{e(g^{r_x}, H(\mathsf{attr}_i)^t)} = e(g, g_2)^{t \cdot q_x(0)}$$

Then the algorithm can decrypt the non-leaf node $x \in \mathcal{T}$ by using polynomial interpolation. Let S_x be the child set of the node x.

$$\begin{aligned} F_x &= \prod_{z \in S_x} F_z^{\Delta_{i,S_x}(0)} \\ &= \prod_{z \in S_x} \left(e(g, g_2)^{t \cdot q_z(0)}\right)^{\Delta_{i,S_x}(0)} \\ &= e(g, g_2)^{t \cdot \sum_{z \in S_x} q_z(0) \cdot \Delta_{i,S_x}(0)} \\ &= e(g, g_2)^{t \cdot q_x(0)} \end{aligned}$$

Since $\mathcal{T}(\gamma) = 1$, the algorithm can decrypt the root node that

$$F_{root} = e(g, g_2)^{t \cdot q_x(0)} = e(g, g_2)^{x_1 t} = e(g, g)^{x_1 x_2 t} = e(g, g)^{\alpha t}$$

The algorithm sets F_{root} as the output of sub-algorithm $\mathcal{D}$. Finally, the algorithm computes the message $M = C_0 / F_{root}$ and return M.

As shown above, the correctness has been shown in the description of the decryption algorithm.

Theorem 5.6 (Correctness)**.** *The above KP-ABE scheme is correct.*

We also show that the above scheme has the linearity property required by LET.

Theorem 5.7 (Linearity)**.** *The sub-algorithm $\mathcal{D}$ has linearity that*

$$\forall s \in \mathbb{Z}_p, \mathcal{D}(\mathsf{SK}^s, C_1, C_2, \{C_i\}_{\mathsf{attr}_i \in \gamma}) = \mathcal{D}(\mathsf{SK}, C_1, C_2, \{C_i\}_{\mathsf{attr}_i \in \gamma})^s$$

Proof. For the decryption of leaf nodes, the computation becomes

$$F'_x = \frac{e(D_x^s, C_1)}{e(R_x^s, C_i)} = \left(\frac{e(D_x, C_1)}{e(R_x, C_i)}\right)^s = F_x^s.$$

For the decryption of non-leaf nodes, the computation becomes

$$F'_x = \prod_{z \in S_x} {F'_z}^{\Delta_{i,S_x}(0)} = \prod_{z \in S_x} F_z^{s\Delta_{i,S_x}(0)} = \left(\prod_{z \in S_x} F_z^{\Delta_{i,S_x}(0)}\right)^s = F_x^s$$

Thus $F'_{root} = F^s_{root}$. □

5.7.2 Construction from the Base Scheme

In this subsection, we apply the LEKS conversion as follows with some key notes.

– $(\mathsf{MSK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(1^\lambda)$: Although the hash functions in the LEKS scheme and the KP-ABE scheme have the same domain and the same codomain, those hash functions cannot be merged since they will be programmed into two different random oracles.

$$\mathbb{G}_1 = \langle g \rangle, \quad e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2, \quad H_1 : \{0,1\}^* \to \mathbb{G}_1, \quad H_2 : \{0,1\}^* \to \mathbb{G}_1,$$
$$x_1, x_2 \in_R \mathbb{Z}_p, \quad g_1 = g^{x_1}, \quad g_2 = g^{x_2},$$
$$\mathsf{SK} = x_1, \quad \mathsf{PK} = (\mathbb{G}_1, \mathbb{G}_2, e, g, g_1, g_2, H_1, H_2)$$

return $(\mathsf{SK}, \mathsf{PK})$.

– $\mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, I_S)$: The key generation algorithm is exactly the same as in the KP-ABE scheme.

$$\forall x \in \text{leaves}(\mathcal{T}), \quad r_x \in_R \mathbb{Z}_p, \quad D_x = g_2^{q_x(0)} H_2(\mathsf{attr}(x))^{r_x}, \quad R_x = g^{r_x}$$

return $\mathsf{SK} = (\mathcal{T}, \{(D_x, R_x)\}_{x \in \text{leaves}(\mathcal{T})})$.

– $C \leftarrow \mathsf{LEKS}(\mathsf{PK}, I_C, W)$:

$$r_1, r_2 \in_R \mathbb{Z}_p, \quad C_1 = g_2^{r_2} H_1(W)^{r_1}, \quad C_2 = g_1^{r_1}, \quad C_3 = g^{r_2},$$
$$\forall \mathsf{attr}_i \in \gamma, \quad C_i = H_2(\mathsf{attr}_i)^{r_2}$$

return $C = (C_1, C_2, C_3, \gamma, \{C_i\}_{\mathsf{attr}_i \in \gamma})$.

– $T \leftarrow \mathsf{Trapdoor}(\mathsf{SK}, W)$:

$$s \in_R \mathbb{Z}_p^+, \quad T_1 = g_1^s, \quad T_2 = H_1(W)^s,$$
$$\forall x \in \text{leaves}(\mathcal{T}), \quad T_{x,1} = D_x^s, \quad T_{x,2} = R_x^s$$

return $T = (T_1, T_2, \mathcal{T}, \{(T_{x,1}, T_{x,2})\}_{x \in \text{leaves}(\mathcal{T})})$.

– $1/0 \leftarrow \mathsf{Test}(C, T)$: The algorithm follows the decryption algorithm in the KP-ABE scheme. It checks whether $\mathcal{T}(\gamma) = 1$ or not. If $\mathcal{T}(\gamma) = 0$, the algorithm returns $\perp$. Otherwise, it continue to compute as in the decryption algorithm. For leaf node $x \in \mathcal{T}$, it computes

$$F_x = \frac{e(T_{x,1}, C_1)}{e(T_{x,2}, C_i)}$$

For non-leaf node, it computes exactly the same as in the decryption algorithm using polynomial interpolation. Eventually, the algorithm computes F_{root} and returns $e(C_1, T_1) \stackrel{?}{=} e(C_2, T_2) \cdot F_{root}$.

5.7.3 Security Proof

The above converted KP-ABKS scheme is similar to Zheng et al.'s KP-ABKS scheme [ZXA14]. The only difference between two schemes is that they use $g_2 g^{b \cdot H(W)}$ as the hash function for the attributes while we use $H_2 : \{0,1\}^* \rightarrow \mathbb{G}_1$.

However, there are some issues in the security proof given in [ZXA14]. Before the simulation provided by the challenger, the adversary selects a target set of attributes Attr^*. In the simulation, the adversary is allowed to query the token generation oracle $\mathcal{O}_{\mathsf{TokenGen}}(T, W)$ with any keyword W other than the target keywords w_0, w_1 and any policy T that $F(\mathsf{Attr}^*, T) = 1$ (i.e. T accepts Attr^*). Stepping into the oracle $\mathcal{O}_{\mathsf{TokenGen}}(T, W)$, the challenger always runs the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}(T)$ to get the secret key sk, and then uses it to generate the requested trapdoor. Since the oracle $\mathcal{O}_{\mathsf{KeyGen}}(T)$ always aborts when $F(Attr^*, T) = 1$, the oracle $\mathcal{O}_{\mathsf{TokenGen}}(T, W)$ always aborts when the adversary does the queries mentioned above. This renders the proof invalid and thus the security of Zheng et al.'s KP-ABKS scheme is unknown.

We prove our KP-ABKS is secure under the ℓ-DCBDH assumption instead of the standard DLIN assumption.

Theorem 5.8. *The proposed KP-ABKS scheme is IND-sCKA (Definition 5.6) secure. If an adversary $\mathcal{A}$ can win the IND-sCKA game (Fig. 5.1) with the advantage ε, an algorithm $\mathcal{S}$ can be constructed to solve ℓ-DCBDH problem (Definition 5.3) in polynomial time with the advantage $\varepsilon' \geq \frac{\varepsilon}{2e(q+2)}$, querying $\mathcal{O}_{\mathsf{Trapdoor}}$ for at most q times where $q \leq \ell$.*

Proof. Let $\delta = (g, g^a, g^b, h, h^c, h^d, \{(g^{f_i}, g^{af_i}, h^{f_i}, h^{af_i})\}_{i=1\ldots\ell}, Z)$ be an ℓ-DCBDH instance (Definition 5.3) to be challenged to the simulator $\mathcal{S}$ for distinguishing that $\delta \in \mathcal{D}_{\text{DCBDH}}$ or $\delta \in \mathcal{D}_{\text{rand}}$. The simulator $\mathcal{S}$ gets two groups $\mathbb{G}_1$, $\mathbb{G}_2$ with a bilinear map e from the instance δ, and sets $g_1 = h$, $g_2 = g^a$. Before the simulator passes $\mathsf{PK} = (\mathbb{G}_1, \mathbb{G}_2, e, g, g_1, g_2, H_1, H_2)$ to the adversary $\mathcal{A}$, it receives a target attribute set Γ from the adversary $\mathcal{A}$. Besides that, the simulator $\mathcal{S}$ simulates the following four oracles.

– $\mathcal{O}_{H_1}$: The hash function H_1 is viewed as a random oracle. Upon requesting the hash value of a keyword W_i, the simulator $\mathcal{S}$ randomly tosses a coin $c_i \in_R \{0, 1\}$ such that $\Pr[c_i = 0] = \alpha$ where the value of α is determined later. The simulator $\mathcal{S}$ also randomly chooses $a_i \in_R \mathbb{Z}_p^+$, and computes the hash value

$$h_i = \begin{cases} h^c h^{a_i} & \text{if } c_i = 0 \\ h^{a_i} & \text{if } c_i = 1 \end{cases}$$

Due to the randomness a_i, the distribution of $\{h_i\}$ is indistinguishable from a random distribution of $\mathbb{G}_1$. Finally, the simulator $\mathcal{S}$ returns the hash value h_i to the adversary $\mathcal{A}$. At the same time, the simulator $\mathcal{S}$ maintains hash list $\mathcal{H} = \{W_i, c_i, a_i, h_i\}$. If the requested keyword W_i is already in the list $\mathcal{H}_1$, the simulator $\mathcal{S}$ returns the corresponding h_i to the adversary $\mathcal{A}$ directly.

– $\mathcal{O}_{H_2}$: The hash function H_2 is viewed as a random oracle. Upon requesting the hash value of an attribute attr_i, the simulator $\mathcal{S}$ randomly chooses $u_i, v_i \in_R \mathbb{Z}_p$, and computes the hash value

$$h'_i = \begin{cases} g^{v_i} & \text{if } \mathsf{attr}_i \in \Gamma \\ g_2^{u_i} g^{v_i} & \text{if } \mathsf{attr}_i \notin \Gamma \end{cases}$$

For $\mathsf{attr}_i \in \Gamma$, the hash value can be viewed as $h'_i = g_2^{u_i} g^{v_i}$ where $u_i = 0$. Due to the randomness u_i and v_i, the distribution of $\{h'_i\}$ is indistinguishable from a random distribution of $\mathbb{G}_1$. Finally, the simulator $\mathcal{S}$ returns the hash value

h'_i to the adversary $\mathcal{A}$. At the same time, the simulator $\mathcal{S}$ maintains hash list $\mathcal{H}_2 = \{\mathsf{attr}_i, u_i, v_i, h'_i\}$. If the requested attribute attr_i is already in the list $\mathcal{H}_2$, the simulator $\mathcal{S}$ returns the corresponding h'_i to the adversary $\mathcal{A}$ directly.

– $\mathcal{O}_{\mathsf{KeyGen}}$: Upon requesting the secret key of the identity with the access tree $\mathcal{T}$, the simulator $\mathcal{S}$ checks whether $\mathcal{T}(\Gamma) = 1$. If $\mathcal{T}(\Gamma) = 1$, the simulator $\mathcal{S}$ aborts since the adversary $\mathcal{A}$ is not allowed to do so. Otherwise, the simulator $\mathcal{S}$ prepares the following two procedures:

  > $\mathsf{PolySat}(\mathcal{T}_x, \gamma, \lambda_x)$: This procedure is to set up polynomials for the access tree $\mathcal{T}_x$ with the secret λ_x where $\mathcal{T}_x(\gamma) = 1$. As the same as the KeyGen algorithm, the simulator $\mathcal{S}$ sets $q_x(0) = \lambda_x$ and randomly chooses other $k_x - 1$ coefficients to complete the polynomial where k_x is the threshold value for that node. To set up polynomials for each child node z of the node x, the simulator calls $\mathsf{PolySat}(\mathcal{T}_z, \gamma, q_x(\mathsf{index}(z)))$.

  > $\mathsf{PolyUnsat}(\mathcal{T}_x, \gamma, g^{\lambda_x})$: This procedure is to set up polynomial for the access tree $\mathcal{T}_x$ with the element g^{λ_x} where $\mathcal{T}_x(\gamma) = 0$ and λ_x is unknown. The simulator $\mathcal{S}$ implicitly defines the polynomial q_x of degree d_x by defining g^{q_x}. At first, the simulator $\mathcal{S}$ sets $g^{q_x(0)} = g^{\lambda_x}$. To complete the definition of the polynomial q_x, d_x more points are required. For the child node z of the node x that $\mathcal{T}_z(\gamma) = 1$, the simulator $\mathcal{S}$ randomly chooses $\lambda_z \in_R \mathbb{Z}_p$ and sets $q_x(\mathsf{index}(z)) = \lambda_z$. Let h_x be the total number of the child node z that $\mathcal{T}_z(\gamma) = 1$. Since $\mathcal{T}_x(\gamma) = 0$, we have $h_x \leq d_x$. Then the simulator $\mathcal{S}$ randomly chooses $d_x - h_x$ points to complete the polynomial q_x. Finally, the simulator can compute $g^{q_x(\mathsf{index}(z))}$ for any child node z by polynomial interpolation. To set up polynomials for each child node z of the node x, the simulator calls $\mathsf{PolySat}(\mathcal{T}_z, \gamma, q_x(\mathsf{index}(z)))$ if $\mathcal{T}_z(\gamma) = 1$, and $\mathsf{PolyUnsat}(\mathcal{T}_z, \gamma, g^{q_x(\mathsf{index}(z))})$ if $\mathcal{T}_z(\gamma) = 0$.

  The simulator $\mathcal{S}$ calls the procedure $\mathsf{PolyUnsat}(\mathcal{T}, \Gamma, h)$. For each leaf node x of $\mathcal{T}$, the simulator $\mathcal{S}$ gets $q_x(0)$ if $\mathsf{attr}(x) \in \Gamma$, or $g^{q_x(0)}$ if $\mathsf{attr}(x) \notin \Gamma$. Finally, the simulator $\mathcal{S}$ computes the secret value for each leaf node x:

  > If $\mathsf{attr}(x) \in \Gamma$, the simulator $\mathcal{S}$ computes

$$\begin{aligned} r_x &\in_R \mathbb{Z}_p^+ \\ D_x &= g_2^{q_x(0)} H_2(\mathsf{attr}(x))^{r_x} \\ R_x &= g^{r_x} \end{aligned}$$

  > If $\mathsf{attr}(x) \notin \Gamma$, the simulator $\mathcal{S}$ queries $\mathcal{O}_{H_2}$ for the entry $(\mathsf{attr}_i, u_i, v_i, h'_i)$

where $\mathsf{attr}_i = \mathsf{attr}(x)$. Then it computes

$$\begin{aligned} r_x &\in_R \mathbb{Z}_p^+ \\ D_x &= \left(g^{q_x(0)}\right)^{\frac{-v_i}{u_i}} h_i'^{r_x} \\ R_x &= \left(g^{q_x(0)}\right)^{\frac{-1}{u_i}} g^{r_x} \end{aligned}$$

For better understanding, we further expand D_x and R_x as follows.

$$\begin{aligned} D_x &= \left(g^{q_x(0)}\right)^{\frac{-v_i}{u_i}} h_i'^{r_x} \\ &= (g^{v_i})^{\frac{-q_x(0)}{u_i}} (g_2^{u_i} g^{v_i})^{r_x} \\ &= g_2^{q_x(0)} (g_2^{u_i} g^{v_i})^{\frac{-q_x(0)}{u_i}} (g_2^{u_i} g^{v_i})^{r_x} \\ &= g_2^{q_x(0)} (g_2^{u_i} g^{v_i})^{r_x - \frac{q_x(0)}{u_i}} \\ &= g_2^{q_x(0)} H_2(\mathsf{attr}(x))^{r_x - \frac{q_x(0)}{u_i}} \\ R_x &= \left(g^{q_x(0)}\right)^{\frac{-1}{u_i}} g^{r_x} \\ &= g^{r_x - \frac{q_x(0)}{u_i}} \end{aligned}$$

From above expansion, we can find that the value pair (D_x, R_x) is valid where the randomness is $r_x - \frac{q_x(0)}{u_i}$ instead of r_x.

– $\mathcal{O}_{\mathsf{Trapdoor}}$: Since the adversary $\mathcal{A}$ can always query $\mathcal{O}_{\mathsf{KeyGen}}$ for the secret key of $\mathcal{T}$, which $T(\Gamma) = 0$, and invoke the real $\mathsf{Trapdoor}$ algorithm to generate trapdoors, we assume that the adversary $\mathcal{A}$ only ask the oracle $\mathcal{O}_{\mathsf{Trapdoor}}$ when $T(\Gamma) = 1$. To generate a trapdoor for the keyword W_i, the simulator $\mathcal{S}$ invokes $\mathcal{O}_{H_1}$ with the keyword W_i. If the corresponding $c_i = 0$, the simulator $\mathcal{S}$ **aborts**. Otherwise, the simulator $\mathcal{S}$ obtains $h_i = h^{a_i}$ from the list $\mathcal{H}_1$. Then the simulator $\mathcal{S}$ gets a tuple $(g^{f_q}, g^{af_q}, h^{f_q}, h^{af_q})$ from the instance δ for the q-th query where $q \leq \ell$. After that, the simulator $\mathcal{S}$ computes the trapdoor T. Since the trapdoor generation requires the secret key part, we first review the real key generation algorithm KeyGen.

$$\begin{aligned} &\forall x \in \text{leaves}(\mathcal{T}), \\ &\quad r_x \in_R \mathbb{Z}_p^+ \\ &\quad D_x = g_2^{q_x(0)} H_2(\mathsf{attr}(x))^{r_x} \\ &\quad R_x = g^{r_x} \end{aligned}$$

The components $q_x(0)$ are the shares of the secret key x_1 that $q_{\text{root}}(0) = x_1$. However, the simulator $\mathcal{S}$ cannot computes those shares since $\mathcal{S}$ does not know

x_1 (recall that $g_1 = g^{x_1} = h$). Instead, the simulator $\mathcal{S}$ calls the procedure $\mathsf{PolySat}(\mathcal{T}, \Gamma, 1)$ to get the shares $q'_x(0)$ where $q'_{\text{root}}(0) = 1$. Due to the linear property, we have $x_1 q'_x(0) = q_x(0)$. Let f_q be the randomness s used in the algorithm $Trapdoor$. The simulator $\mathcal{S}$ queries $\mathcal{O}_{H_2}$ for the entry $(\mathsf{attr}_i, u_i, v_i, h'_i)$ where $\mathsf{attr}_i = \mathsf{attr}(x)$, and calculates

$$
\begin{aligned}
T_1 &= g_1^s = h^{f_q} \\
T_2 &= H_1(W)^s = (h^{a_i})^{f_q} = (h^{f_q})^{a_i} \\
&\forall x \in \text{leaves}(\mathcal{T}), \\
&\quad r_x \in_R \mathbb{Z}_p^+ \\
&\quad T_{x,1} = D_x^s \\
&\qquad = \left(g_2^{q_x(0)} H_2(\mathsf{attr}(x))^{r_x} \right)^s \\
&\qquad = \left((g^a)^{x_1 q'_x(0)} H_2(\mathsf{attr}(x))^{r_x} \right)^s \\
&\qquad = \left((g^{x_1})^{a q'_x(0)} H_2(\mathsf{attr}(x))^{r_x} \right)^s \\
&\qquad = \left(h^{a q'_x(0)} (g_2^{u_i} g^{v_i})^{r_x} \right)^{f_q} \\
&\qquad = (h^{a f_q})^{q'_x(0)} (g_2^{u_i} g^{v_i})^{r_x})^{f_q} \\
&\qquad = (h^{a f_q})^{q'_x(0)} (g^{a f_q})^{u_i r_x} (g^{f_q})^{v_i r_x} \\
&\quad T_{x,2} = R_x^s \\
&\qquad = (g^{r_x})^{f_q} \\
&\qquad = (g^{f_q})^{r_x}
\end{aligned}
$$

Finally, the simulator $\mathcal{S}$ returns $T = (T_1, T_2, \mathcal{T}, \{(T_{x,1}, T_{x,2})\}_{x \in \text{leaves}(\mathcal{T})})$ to the adversary $\mathcal{A}$.

At some point, the adversary $\mathcal{A}$ outputs two target keywords W_0 and W_1. Then the simulator $\mathcal{S}$ invokes $\mathcal{O}_{H_1}$ for the hash value of W_0 and W_1. If the corresponding c_0, c_1 to the keywords W_0 and W_1 is equal to 1, the simulator $\mathcal{S}$ **aborts**. Otherwise, the simulator $\mathcal{S}$ randomly selects $b \in_R \{0, 1\}$ such that $c_b = 0$. In other words, the simulator $\mathcal{S}$ always has $H(W_b) = h^c h^{a_i}$. Before computing the ciphertext $C = (C_1, C_2, C_3)$, we observe the genuine LEKS algorithm. Let (d, b) be the randomnesses (r_1, r_2) used in the algorithm LEKS. We have

$$
\begin{aligned}
C_1 &= g_2^{r_2} H(W)^{r_1} = g^{ab} (h^c h^{a_i})^d = g^{ab} h^{cd} \cdot (h^d)^{a_i} \\
C_2 &= g_1^{r_1} = h^d \\
C_3 &= g^{r_2} = g^b \\
\forall \mathsf{attr}_i \in \Gamma, \quad C_i &= H_2(\mathsf{attr}_i)^{r_2} = (g^{v_i})^b = (g^b)^{v_i}
\end{aligned}
$$

The simulator $\mathcal{S}$ replaces $g^{ab}h^{cd}$ with Z in the instance δ: $C_1 = Z \cdot (h^d)^{a_i}$. The resulted ciphertext C is consistent if and only if $Z = g^{ab}h^{cd}$. After received the ciphertext C from the simulator $\mathcal{S}$, the adversary $\mathcal{A}$ can continue to query all the oracles as before with defined restrictions in IND-sCKA game. Eventually, the adversary $\mathcal{A}$ outputs a bit b'. If $b = b'$, the ciphertext C is believed consistent that $Z = g^{ab}h^{cd}$, and the simulator $\mathcal{S}$ outputs $\delta \in \mathcal{D}_{\text{DCBDH}}$. Otherwise, the simulator $\mathcal{S}$ outputs $\delta \in \mathcal{D}_{\text{rand}}$.

Lemma 5.5. *Let ρ be the probability that the simulator $\mathcal{S}$ does not abort. Assuming the probability of $\delta \in \mathcal{D}_{DCBDH}$ is $\frac{1}{2}$ and the advantage of the adversary $\mathcal{A}$ winning the IND-sCKA game (Fig. 5.1) is ε, the advantage ε' of the simulator $\mathcal{S}$ solving the ℓ-DCBDH problem is at least $\frac{\rho\varepsilon}{2}$.*

Proof. Let E_s be the event that the simulator $\mathcal{S}$ successfully solves the ℓ-DCBDH problem, E_w be the event that the adversary $\mathcal{A}$ wins in the IND-sCKA game (Fig. 5.1) that $\Pr[E_w] = \frac{1}{2} + \varepsilon$, E_a be the event that $\mathcal{S}$ aborts, and E_r be the event that $\mathcal{S}$ solves the problem with random guess that $\Pr[E_r] = \frac{1}{2}$. If $\delta \in \mathcal{D}_{\text{rand}}$, the behaviour of the adversary $\mathcal{A}$ is unpredictable. Therefore, the adversary $\mathcal{A}$ wins the IND-sCKA game (Fig. 5.1) better than a random guess that $\Pr[E_w] \geq \Pr[E_r]$. Thus we have the probability that $\mathcal{S}$ successfully solves the ℓ-DCBDH problem.

$$\begin{aligned}
\Pr[E_s] &= \frac{1}{2}\Pr[E_s \mid \delta \in \mathcal{D}_{\text{DCBDH}}] + \frac{1}{2}\Pr[E_s \mid \delta \in \mathcal{D}_{\text{rand}}] \\
&\geq \frac{1}{2}\left(\Pr[E_w]\Pr[\neq E_a] + \Pr[E_r]\Pr[E_a]\right) + \frac{1}{2}\Pr[E_r] \\
&= \frac{1}{2}\left((\frac{1}{2} + \varepsilon)\rho + \frac{1}{2}(1-\rho)\right) + \frac{1}{2}\cdot\frac{1}{2} \\
&= \frac{1}{2}\rho\varepsilon + \frac{1}{2}
\end{aligned}$$

Finally, we calculate the advantage ε' as follows.

$$\varepsilon' = \left|\varepsilon - \frac{1}{2}\right| \geq \frac{1}{2}\rho\varepsilon$$

□

Lemma 5.6. *The probability ρ that the simulator $\mathcal{S}$ does not abort is at least $\frac{1}{e(q+2)}$ with q times of $\mathcal{O}_{\mathsf{Trapdoor}}$ queries.*

Proof. There are two possible points that the simulator $\mathcal{S}$ may abort.

1. The simulator $\mathcal{S}$ aborts in answering $\mathcal{O}_{\mathsf{Trapdoor}}$ if $c_i = 0$. The probability $\Pr[E_1]$ of $\mathcal{S}$ not aborting is $\Pr[E_1] = (1-\alpha)^q$.

2. The simulator $\mathcal{S}$ aborts in the challenge phase if $c_0 = c_1 = 1$. The probability $\Pr[E_2]$ of $\mathcal{S}$ not aborting if $\Pr[E_2] = 1 - (1-\alpha)^2 = \alpha(1-\alpha)$.

Since all events are independent, we have

$$\rho = \Pr[E_1] \cdot \Pr[E_2] = (1-\alpha)^q \cdot \alpha(1-\alpha) = \alpha(1-\alpha)^{q+1}.$$

The value of ρ is maximised when $\alpha = \frac{1}{q+2}$. Therefore, we have

$$\begin{aligned}
\rho &= \frac{1}{q+2}\left(1 - \frac{1}{q+2}\right)^{q+1} \\
&\geq \frac{1}{q+2} \lim_{q\to\infty} \left(1 - \frac{1}{q+2}\right)^{q+1} \\
&= \frac{1}{e(q+2)}
\end{aligned}$$

□

Combining the above two lemmas, we have

$$\varepsilon' \geq \frac{1}{2}\rho\varepsilon \geq \frac{\varepsilon}{2e(q+2)}$$

□

5.8 Chapter Summary

In this chapter, we introduced a Decisional Bilinear (P, f)-Diffie-Hellman problem family and demonstrated its hardness under the generic bilinear group model. We also derived a hard computational problem named Decisional ℓ-Combined Bilinear Diffie-Hellman problem from the (P, f)-DBDH problem family. As the main contribution of this chapter, we proposed Linear Encryption with Keyword Search and its security model, and defined Linear Encryption Template which can be used to convert encryption schemes into the corresponding LEKS schemes. To show concrete instances of our LEKS conversion framework, we converted a PKE scheme into a PEKS scheme, a KP-ABE scheme into a KP-ABKS scheme and proved their security in the random oracle model, assuming the hardness of the ℓ-DCBDH problem.

Our future work will be finding more LET-compatible encryption schemes, converting them into searchable schemes and proving their security.

Part III

Controlled Information Revelation

Chapter 6

Achieving IND-CCA Security for Functional Encryption for Inner Products

Functional Encryption (FE) allows the authorised parties to reveal partial information of the plaintext hidden in a ciphertext while in conventional encryption decryption is all-or-nothing. Focusing on the functionality of inner product evaluation (i.e. given vectors $\vec{x}$ and $\vec{y}$, calculate $\langle \vec{x}, \vec{y} \rangle$), Abdalla et al. [ABDCP15] proposed a Functional Encryption for Inner Products (FE-IP) scheme with Selective Security against Chosen Plaintext Attacks (s-IND-CPA). In some recent works by Abdalla et al. [ABCP16] and Agrawal et al. [ALS16], FE-IP schemes with the security of Indistinguishability under Chosen Plaintext Attacks (IND-CPA) have also been proposed. In order to achieve Indistinguishability under adaptive Chosen Ciphertext Attacks (IND-CCA) for FE-IP, in this chapter, we propose a generic construction of FE-IP from Hash Proof System (HPS). We prove the constructed FE-IP scheme is IND-CCA secure, assuming the hardness of the Subset Membership Problem (SMP). In addition, we give two instantiations of our generic construction from the Decisional Diffie-Hellman (DDH) and Decisional Composite Residuosity (DCR) assumptions. Parts of this work appeared in [ZMY17].

6.1 Introduction

Encryption provides information confidentiality such that messages are hidden and can only be revealed by authorised parties. In traditional encryption, accessing to the plaintext is in an all-or-nothing manner. Precisely, Alice encrypts a message using Bob's encryption key and sends the ciphertext to Bob. Later, Bob can decrypt the ciphertext to read the message using his decryption key while a malicious interceptor Eve gets no information about the encrypted message. Whereas in Functional Encryption (FE), it is possible for different authorised parties to reveal different partial information of the plaintext from a ciphertext by granting them different secret keys. It is also possible to control the information leaked from the ciphertexts. In detail, a FE enables the authorised receivers to reveal the output of a functionality $f(k, x)$ from a ciphertext containing the plaintext x and a secret key

associated with a function key value k.

In this chapter, we focus on the functional encryption for the functionality of inner product evaluation where $f(\vec{x}, \vec{y}) = \langle \vec{x}, \vec{y} \rangle$. A direct application of such a FE scheme is privacy-preserving descriptive statistics such as calculating the weighted mean or sum of a list of integers. For instance, suppose in a high school the subject grades of each student are stored in a vector $\vec{y}$ which is encrypted under the school manager Alice's public key. As a university admission officer, Bob wants to offer scholarship to those students who are excellent at mathematics, physics, English, and good at other subjects. To ensure good students can get the scholarship, Alice decides to assist Bob in identifying the candidates. At the same time, Alice does not want to reveal the grades of all the students to Bob for privacy reasons. With Functional Encryption for Inner Products (FE-IP), Alice can generate a secret key for Bob, which is associated with a vector $\vec{x} = (10, 8, 8, 5, 5)$ that represents the weight for different subjects (i.e. 10 for mathematics, 8 for physics and English, and 5 for other subjects). Later, Bob can run the decryption algorithm to get the weighted sum of each student's subject grades, and nothing else. For example, Charlie has a grade vector $\vec{y} = (90, 70, 80, 50, 60)$. The function $f(\vec{x}, \vec{y})$ gives $\langle \vec{x}, \vec{y} \rangle = 10 \times 90 + 8 \times 70 + 8 \times 80 + 5 \times 50 + 5 \times 60 = 2650$. In this case, Bob can only learn the result 2650 but nothing else about $\vec{y}$.

The security of FE-IP is defined by the notion of Indistinguishability under Chosen Plaintext Attacks (IND-CPA) in [ABDCP15]. However, such a security notion is not strong enough to cover the following variation of the above scenario. In order to restrict Bob's ability to calculate the weighted sum of each student, Alice gives the security key for the vector $\vec{x}$ to her colleague David instead of directly giving it to Bob. Consequently, Bob can only get the result of $f(\vec{x}, \vec{y})$ from David by sending the ciphertext to him, and David can reject any queries related to those students whose grades are below a threshold. If the scheme is malleable, then Bob can modify a rejected ciphertext such that it can pass the threshold.

In this chapter, we aim to build FE-IP schemes with the security of Indistinguishability under adaptive Chosen Ciphertext Attacks (IND-CCA), which is stronger than IND-CPA and can withstand the attack described above.

6.1.1 Related Work

The notion of Functional Encryption is introduced by Lewko et al. [LOS+10] and later formally defined by Boneh et al. [BSW11]. In [BSW11], the security of functional encryption is naturally defined via *indistinguishability*-based security (IND-security) where an adversary cannot distinguish which message x_0 or x_1 is encrypted in the ciphertext with oracles provided according to the attacking model.

However, the IND-security is not sufficient for the general functional encryption [BSW11, O'N10], and thus simulation-based security (SIM-security) has been proposed. However, the SIM-security is only achievable in the programmable random oracle model.

For generic functionality, Goldwasser et al. [GKP+13] proposed a FE scheme for circuits. Later in [GGG+14], the functionality is further extended to accept multiple inputs such that it is able to compute $f(k, x_1, \ldots, x_n)$ instead of $f(k, x)$. Since the construction for generic functionalities is very inefficient for practical use, the construction for specific functionality has been the main focus. It is worth noting there is a subclass of FE named Predicate Encryption (PE) [KSW08]. Its message space X consists of two subspaces, index space I and payload space M. For a predicate $P : K \times I \to \{0, 1\}$, the functionality $f : K \times X \to M \cup \{\perp\}$ is defined as $f(k, (ind, m)) = m$ if $P(k, ind) = 1$ or $f(k, (ind, m)) = \perp$ otherwise. More subclasses of the functionalities can be derived from the PE class, including but not limited to Identity-Based Encryption (IBE) [BF01], Key-Policy Attribute-Based Encryption (KP-ABE) [OT12, SW05], Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [BSW07], Hidden Vector Encryption (HVE) [BW07], Inner Product Encryption (IPE) [LOS+10, OT12], and Deterministic Finite Automata (DFA) Based Encryption (Functional Encryption for Regular Languages) [TCL14, Wat12]. Another notable FE is searchable encryption [BDCOP04] where $f(k, x) = 1$ if $k = x$ or $f(k, x) = 0$ otherwise where k and x are the keywords embedded in the trapdoor and ciphertext, respectively. For further discussions on the functionalities, we refer the reader to Section 3.3.

Recently, Abdalla et al. [ABDCP15] investigated a new functionality $f(\vec{x}, \vec{y}) = \langle \vec{x}, \vec{y} \rangle$, i.e. to calculate the inner product of two vectors $\vec{x}$ and $\vec{y}$ where $\vec{x}$ is embedded in the secret key and $\vec{y}$ is embedded in the ciphertext. Unlike Inner Product Encryption [LOS+10, OT12] where inner product is used for access control, the new functionality here is to compute the actual inner product value. In [ABDCP15], Abdalla et al. proposed a FE-IP scheme, which has Selective Security against Chosen Plaintext Attacks (s-IND-CPA). The scheme is generic as it can be constructed from any s-IND-CPA secure public key encryption, which is secure under randomness reuse and has linear key homomorphism and linear ciphertext homomorphism under shared randomness. Based on the generic construction, two instantiations are given from Decisional Diffie-Hellman (DDH) assumption and Learning With Error (LEW) assumption respectively. In some recent works [ABCP16, ALS16], FE-IP schemes with IND-CPA security were also proposed. Specifically, Abdalla et al.[ABCP16] proposed another generic construction with IND-CPA security from any s-IND-CPA secure public key encryption with the same requirements as in [ABDCP15]. They also showed that the IND-CPA security and Non-Adaptive Simulation (NA-SIM)

security are equivalent for inner product functionality. Furthermore, an instantiation from Decisional Composite Residuosity (DCR) assumption is also proposed in [ABCP16].

In addition to the confidentiality, a notion called *function privacy* has also been investigated for FE which means an adversary should not be able to distinguish k (or f_k as $f(k, x) = f_k(x)$) from a secret key SK_k. For DFA-based encryption, Tseng et al. [TCL14] proposed a version with function privacy. For FE-IP, Bishop et al. [BJK15] proposed a scheme with function privacy. However, the schemes [BJK15, TCL14] with function privacy are proposed in the private/symmetric key setting while normal FE schemes [ABCP16, ABDCP15, Wat12] are in the public key setting.

6.1.2 Our Contribution

In this chapter, we define the notion of Indistinguishability under adaptive Chosen Ciphertext Attacks (i.e. IND-CCA, or more precisely IND-CCA2 security) for the general Functional Encryption. We also present the precise definition of Functional Encryption for Inner Products. In particular, we show that the secret keys for the functions $\langle \vec{x}_1, \cdot \rangle, \cdots, \langle \vec{x}_n, \cdot \rangle$ implies the secret key for the function $\langle \vec{x}', \cdot \rangle$ where $\vec{x}' \in \text{span}(\vec{x}_1, \ldots, \vec{x}_n)$.

As the main contribution of this chapter, we propose under certain conditions an IND-CCA secure Functional Encryption for Inner Products (FE-IP) scheme from Hash Proof System (HPS), assuming the hardness of the Subset Membership Problem (SMP). In the generic construction, we require two HPSs Ξ_1 and Ξ_2 with some special properties as the building blocks. In detail, Ξ_1 is required to be *diverse* (Definition 6.5) and have *key linearity* (Definition 6.3) and *hash linearity* (Definition 6.4). For Ξ_2, we require it to be *universal*$_2$ (Definition 2.21) and have *hash linearity*. We show that those special properties are not hard to achieve. In [CS02], Cramer and Shoup constructed HPSs from a diverse group system $\mathbf{G} = (\mathcal{H}, X, L, \Pi)$. We show that their constructions have the key linearity and the hash linearity. If the hash codomain Π of the underlying diverse group system has prime order, the constructed HPS has the property of *diversity*. In other words, we can generically construct an IND-CCA secure FE-IP scheme from a diverse group system $\mathbf{G} = (\mathcal{H}, X, L, \Pi)$ when $|\Pi|$ is prime.

In addition, we propose a concrete IND-CCA secure FE-IP scheme from DDH assumption as an instantiation of our generic construction. Note that if we remove the Non-Interactive Zero Knowledge (NIZK) proof part of construction 6.7, the resulting scheme is exactly the same as the schemes in [ABCP16, ALS16]. Thus the efficiency is the same as [ABCP16, ALS16]. Furthermore, we also instantiate a

IND-CCA secure FE-IP scheme from DCR assumption where the computation of discrete logarithm is easy.

6.1.3 Chapter Organisation

The rest of this chapter is organised as follows. Beginning with Section 6.2, we give out a precise definition of FE-IP and introduce the IND-CCA security model. In Section 6.3, we define new properties of HPS, review the constructions of the HPS, and show that the existing construction has the new defined properties. After that, we propose a generic construction of IND-CCA secure FE-IP with security proof in Section 6.4. In addition, instantiations of our generic construction from DDH and DCR assumptions are provided in Section 6.5. Finally, this chapter is summarised in Section 6.6.

6.2 Formal Definitions

6.2.1 Derived Syntax

Let $\mathbb{G}_x$, $\mathbb{G}_y$, $\mathbb{G}_z$ be three abelian groups where there exists an efficient inner product computation $\langle \cdot, \cdot \rangle : \mathbb{G}_x \times \mathbb{G}_y \to \mathbb{G}_z$. The FE-IP is associated with a functionality $f : \mathbb{G}_x^\delta \times \mathbb{G}_y^\delta \to \mathbb{G}_z$, mapping two δ-dimension vectors into a single group $\mathbb{G}_z$ such that $f(\vec{x}, \vec{y}) = \langle \vec{x}, \vec{y} \rangle = \sum_{i=1}^{\delta} \langle x_i, y_i \rangle$ for all $\vec{x} = (x_1, \ldots, x_\delta) \in \mathbb{G}_x^\delta$ and $\vec{y} = (y_1, \ldots, y_\delta) \in \mathbb{G}_y^\delta$. Based on the functionality f, we derive the syntax of the functional encryption from Definition 3.1.

Definition 6.1 (Functional Encryption for Inner Products)**.** *A Functional Encryption for Inner Products (FE-IP) scheme for a functionality $f : \mathbb{G}_x^\delta \times \mathbb{G}_y^\delta \to \mathbb{G}_z$ consists of the following four polynomial time algorithms:*

- $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\delta)$*: The randomised system setup algorithm takes a security parameter 1^λ and a unary value 1^δ that specifies the maximum vector dimension as input. Then it generates system-wide parameters and a key pair $(\mathsf{PK}, \mathsf{MSK})$.*

- $\mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, \vec{x})$*: The randomised secret key generation algorithm takes a master secret key MSK and a vector $\vec{x} \in \mathbb{G}_x^d$ with dimension d. If $d > \delta$, the extra dimensions of $\vec{x}$ is discarded. If $d < \delta$, the vector $\vec{x}$ is reconstructed to the dimension δ by filling an additive identity element 0 (i.e. $\vec{x}' = (\vec{x}, \underbrace{0, \ldots, 0}_{\delta - d})$). After that, the algorithm generates a secret key SK for the (modified) vector $\vec{x}$ with dimension δ.*

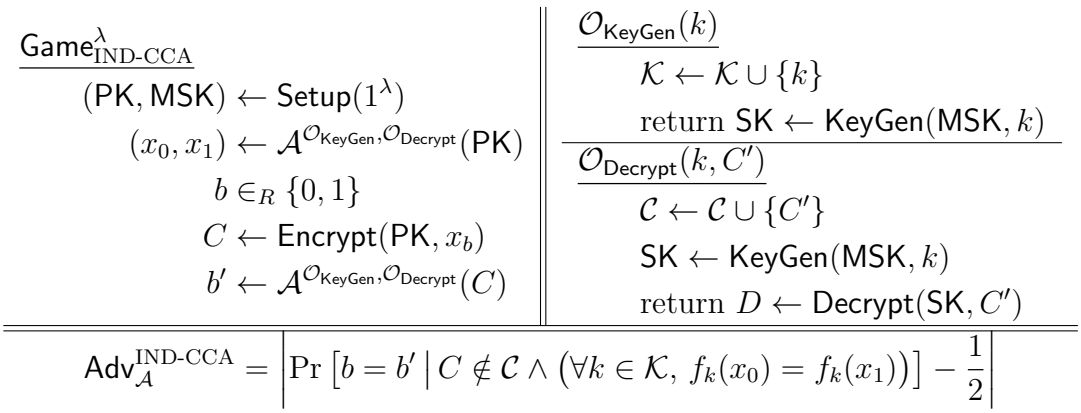

Figure 6.1: IND-CCA Game

- $C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, \vec{y})$*: The randomised encryption takes a public key* PK *and a vector* $\vec{y} \in \mathbb{G}_y^d$ *with dimension* d*. If* $d \neq \delta$*, the ciphertext may still be constructed. However, it may not be decrypted properly. Hence, the same modification to* $\vec{x}$ *in the algorithm* KeyGen *is applied to* $\vec{y}$*. After that, the algorithm generates a ciphertext* C *for the (modified) vector* $\vec{y}$ *with dimension* δ*.*

- $D \leftarrow \mathsf{Decrypt}(\mathsf{SK}, C)$*: The deterministic decryption algorithm takes a secret key* SK *for* $\vec{x}$ *and a ciphertext* C *of* $\vec{y}$*, and computes* $D = \langle \vec{x}, \vec{y} \rangle \in \mathbb{G}_z$*. If the decryption fails, the algorithm outputs a special symbol* $\perp$*.*

6.2.2 Enhanced Security Model

In this chapter, we consider the indistinguishability-based security and enhance the IND-CPA security model (Definition 3.2) defined in [BSW11] to the Indistinguishability under adaptive Chosen Ciphertext Attacks (IND-CCA) as the generalisation of the IND-CCA2 security [RS92] for Public Key Encryption (PKE) schemes [CS98]. The difference is that the decryption oracle $\mathcal{O}_{\mathsf{Decrypt}}$ is not allowed in the IND-CPA game at any stage. The IND-CCA game (Fig. 6.1) is defined as follows where an adaptive adversary $\mathcal{A}$ tries to distinguish a ciphertext from two chosen plaintexts x_0 and x_1.

Setup phase The challenger $\mathcal{S}$ runs $\mathsf{Setup}(1^{\lambda})$ to generate a key pair $(\mathsf{MSK}, \mathsf{PK})$, and passes the public key PK to the adversary $\mathcal{A}$.

Pre-challenge phase The adversary $\mathcal{A}$ can adaptively query the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}$ for the secret key SK of a function f_k from the challenger $\mathcal{S}$. The restriction is that $\mathcal{A}$ can only query the secret keys for the functionality f_k such that $f_k(x_0) = f_k(x_1)$ where x_0 and x_1 are the target plaintexts in the challenge phase. Otherwise, the game is trivial since $\mathcal{A}$ can simply win the

game by testing $\mathsf{Decrypt}(\mathsf{SK}_k, C) \stackrel{?}{=} f_k(x_0)$. Besides that, the adversary $\mathcal{A}$ can also ask the challenger $\mathcal{S}$ for decrypting a ciphertext C' of x to obtain the output of $f_k(x)$ for any $k \in K$ via the decryption oracle $\mathcal{O}_{\mathsf{Decrypt}}$.

Challenge phase At some point, the adversary $\mathcal{A}$ outputs two target plaintexts x_0 and x_1. The challenger $\mathcal{S}$ randomly selects a bit $b \in_R \{0, 1\}$, and generates a target ciphertext $C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, x_b)$. Then $\mathcal{S}$ passes C to the adversary $\mathcal{A}$.

Post-challenge phase The adversary $\mathcal{A}$ can continue to query the oracle $\mathcal{O}_{\mathsf{KeyGen}}$ with the same restriction as before, and the oracle $\mathcal{O}_{\mathsf{Decrypt}}$ with the restriction that $\mathcal{A}$ cannot query the target ciphertext C since $\mathcal{A}$ can win the game trivially by testing $\mathcal{O}_{\mathsf{Decrypt}}(k, C) \stackrel{?}{=} f_k(x_0)$ for some $k \in K$ such that $f_k(x_0) \neq f_k(x_1)$.

Guessing phase Eventually, the adversary $\mathcal{A}$ outputs a bit b', and $\mathcal{A}$ wins if $b = b'$.

The advantage of $\mathcal{A}$ winning the IND-CCA game (Fig. 6.1) is

$$\mathsf{Adv}_{\mathcal{A}}^{\text{IND-CCA}} = \left| \Pr\left[b = b' \,\middle|\, C \notin \mathcal{C} \wedge \left(\forall k \in \mathcal{K},\ f_k(x_0) = f_k(x_1)\right)\right] - \frac{1}{2} \right|$$

Definition 6.2 (IND-CCA Security)**.** *A FE scheme is Indistinguishable under adaptive Chosen Ciphertext Attacks (IND-CCA) if* $\mathsf{Adv}_{\mathcal{A}}^{\text{IND-CCA}}$ *is a negligible function for all adversary* $\mathcal{A}$ *winning the the IND-CCA game (Fig. 6.1) in polynomial time.*

Before introducing the security model of FE-IP, we review the inner product functionality along with the vector space first.

Due to the linearity that $\langle x_0 + x_1, y\rangle = \langle x_0, y\rangle + \langle x_1, y\rangle$ for all $x_0, x_1 \in \mathbb{G}_x$ and $y \in \mathbb{G}_y$, we have $\mu\langle \vec{x}, \vec{y}\rangle = \langle \mu\vec{x}, \vec{y}\rangle$ for all $\mu \in \mathbb{Z}$, $\vec{x} \in \mathbb{G}_x^\delta$, and $\vec{y} \in \mathbb{G}_y^\delta$. Thus the ability of the secret key for a vector $\vec{x}$ is not only to calculate $f(\vec{x}, \vec{y})$ but also to compute $f(\mu\vec{x}, \vec{y}) = \langle \mu\vec{x}, \vec{y}\rangle = \mu\langle \vec{x}, \vec{y}\rangle = \mu f(\vec{x}, \vec{y})$, which is equivalent to the ability of the secret key for the vector $\mu\vec{x}$ for all $\mu \in \mathbb{Z}$. In other words, the key generation algorithm KeyGen actually generates a secret key for a vector space $\text{span}(\vec{x})$ linearly spanned by $\vec{x}$ instead of a single vector $\vec{x}$. Generally, given multiple secret keys for a vector set $S = \{\vec{x}_1, \ldots, \vec{x}_n\}$, we are able to compute $f(\vec{x}, \vec{y})$ for all $\vec{x} \in \text{span}(S)$. It is possible since $f(\vec{x}, \vec{y}) = \sum_{i=1}^n \mu_i f(\vec{x}_i, \vec{y})$ where $\vec{x} = \sum_{i=1}^n \mu_i \vec{x}_i$. Notably, if we obtain secret keys for a vector set S such that $\text{span}(S) = \mathbb{G}_x$ (e.g. S contains δ linearly dependent vectors), we have the same ability of the master secret key without compromising it.

Since δ secret keys for linearly independent vectors are equivalent to the master secret key, the function key space K (recall Definition 3.1) is reduced to the size of δ, which is polynomial bounded. Let $\vec{v}_1, \ldots, \vec{v}_\delta$ be a basis of $\mathbb{G}_x$. Intuitively,

one may think that the "brute force" construction in [BSW11] becomes practical by encrypting the output of $f(\vec{v}_1, \vec{y}), \ldots, f(\vec{v}_\delta, \vec{y})$ instead of the vector $\vec{y}$ where the resulting ciphertext size is $\Theta(\delta)$. However, it is not true since $\mathrm{span}(\vec{v}_1 + \vec{v}_2) \neq \mathrm{span}(\vec{v}_1, \vec{v}_2)$ where $\vec{v}_1$ and $\vec{v}_2$ are independent. Hence, a proper construction is still required.

Besides that, if $\mathbb{G}_x = \mathbb{G}_y = \mathbb{G}_z = \mathbb{F}$ are the same field, it is impossible to hide $\vec{x}$ in the public key setting, given the secret key for the vector $\vec{x}$. Since it is in the public key setting, δ linearly independent vectors $\vec{y}_1, \ldots, \vec{y}_\delta$ can be chosen and encrypted freely. By decrypting the above ciphertexts with the secret key for the vector $\vec{x} = (x_1, \ldots, x_\delta)$, we can obtain the results $\{D_i = f(\vec{x}, \vec{y}_i)\}_{i=1\ldots\delta}$. After that, we can calculate $\vec{x}$ by solving the following matrix equation in polynomial time.

$$\begin{bmatrix} \vec{y}_1 \\ \vdots \\ \vec{y}_\delta \end{bmatrix} \vec{x}^\top = \begin{bmatrix} D_1 \\ \vdots \\ D_\delta \end{bmatrix}$$

Hence, it is impossible to achieve *function privacy*. As a side effect, it is "safe" to provide $\vec{x}$ along with the secret key in the key generation algorithm KeyGen.

For the security model, the definition of the IND-CPA security and the IND-CCA security can be derived from the security model of the general functional encryption. The difference is that the setup algorithm is required to take an additional parameter 1^δ.

6.3 Extended Hash Proof Systems

In this section, we extend Hash Proof System (HPS) (see Definition 2.18 in Section 2.5.3) introduced by Cramer and Shoup [CS02] with some extra properties so that we can use it to construct our scheme.

6.3.1 Extra Properties

The original definition of HPS from [CS02] is not sufficient for our schemes, and the following extended properties are required.

Definition 6.3 (Key Linearity)**.** *A HPS is linear key homomorphic if K and S are additive abelian groups and*

$$\forall a, b \in K, \quad \mathsf{PKGen}(a) + \mathsf{PKGen}(b) = \mathsf{PKGen}(a + b) \in S.$$

Particularly, if a HPS has key linearity, we have $\mu \cdot \mathsf{PKGen}(\mathsf{SK}) = \mathsf{PKGen}(\mu \cdot \mathsf{SK})$ for all $\mathsf{SK} \in K$ and $\mu \in \mathbb{Z}$.

Definition 6.4 (Hash Linearity)**.** *A HPS is linear hash homomorphic if K and Π are additive abelian groups and*

$$\forall a, b \in K, \quad \forall x \in X, \quad \mathsf{Hash}(a, x) + \mathsf{Hash}(b, x) = \mathsf{Hash}(a + b, x) \in \Pi.$$

Similar to the key linearity, we have $\mu \cdot \mathsf{Hash}(\mathsf{SK}, x) = \mathsf{Hash}(\mu \cdot \mathsf{SK}, x)$ for all $\mathsf{SK} \in K$, $x \in X$, and $\mu \in \mathbb{Z}$.

Definition 6.5 (Diversity)**.** *A HPS is diverse if there exists $\pi \in \Pi$ such that $\pi \neq 0$ and for all $x \in X \setminus L$, there exists $\mathsf{SK} \in K$ such that $\mathsf{PKGen}(\mathsf{SK}) = 0$ and $\mathsf{Hash}(\mathsf{SK}, x) = \pi$. Formally,*

$$\exists \pi \in \Pi \setminus \{0\}, \quad \forall x \in X \setminus L, \quad \exists \mathsf{SK} \in K, \quad \mathsf{PKGen}(\mathsf{SK}) = 0 \wedge \mathsf{Hash}(\mathsf{SK}, x) = \pi$$

We call such an element π as an element derived from the diversity.

The new properties of key linearity and hash linearity are easy to understand while the diversity is tricky. Therefore, we discuss the relation between the diversity and the smoothness (Definition 2.22).

Recalling the basic properties of HPS in Section 2.5.3, the universal$_2$ property is a generalisation of the universal property, and implies it. Informally, the universal property requires that a public hash key PK does not leak any information on hashing a word not in the language. Furthermore, the smoothness requires not only no information leakage on PK as the universal property but also indistinguishability of the hash value from a random element in the codomain. From [CS02], universal$_2$ HPS and smooth HPS are constructible from universal HPS. The above properties are important since the smooth property provides the IND-CPA security and the universal$_2$ property provides the IND-CCA security in the PKE construction from HPSs.

The natural meaning of the diversity is that the secret hash keys are diverse on the fixed public hash key, resulted in different hash values for the words not in the language. In other words, the diversity enables that the partial information of the hash value of a word $x \in X \setminus L$ is uncertain with the public hash key provided. Formally, we have the following theorem.

Theorem 6.1. *Let Ξ be a HPS with diversity, key linearity, and hash linearity. Let $\xi \in \Pi \setminus \{0\}$ be an element derived from diversity such that $\forall x \in X \setminus L, \exists \mathsf{SK}^* \in K, \mathsf{PKGen}(\mathsf{SK}^*) = 0 \wedge \mathsf{Hash}(\mathsf{SK}^*, x) = \xi$. Let $x \in_R X \setminus L$, $\mathsf{SK} \in_R K$, $\mathsf{PK} = \mathsf{PKGen}(\mathsf{SK})$, $\pi = \mathsf{Hash}(\mathsf{SK}, x)$, and $\pi' \in_R \Pi' = \{a\xi \mid a \in \mathbb{Z}\}$. Given two probability distributions $\mathcal{D}_H = \{(x, \mathsf{PK}, \pi)\}$ and $\mathcal{D}_R = \{(x, \mathsf{PK}, \pi + \pi')\}$, the adversary $\mathcal{A}$*

has no advantage $\mathsf{Adv}_{\mathcal{A}}$ *in distinguishing* $\mathcal{D}_H$ *and* $\mathcal{D}_R$.

$$\mathsf{Adv}_{\mathcal{A}} = |\Pr[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_H)] - \Pr[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_R)]| = 0$$

Proof. Let $r \in \mathbb{Z}$, and $\mathsf{SK}' = \mathsf{SK} + r\mathsf{SK}^*$. Since

$$\mathsf{PKGen}(\mathsf{SK}') = \mathsf{PKGen}(\mathsf{SK} + r\mathsf{SK}^*) = \mathsf{PKGen}(\mathsf{SK}) + r\mathsf{PKGen}(\mathsf{SK}^*) = \mathsf{PK}, \tag{6.1}$$

$$\mathsf{Hash}(\mathsf{SK}', x) = \mathsf{Hash}(\mathsf{SK} + r\mathsf{SK}^*, x) = \mathsf{Hash}(\mathsf{SK}, x) + r\mathsf{Hash}(\mathsf{SK}^*, x) = \pi + r\xi, \tag{6.2}$$

we have there are different secret hash keys SK' corresponding to the same public hash key PK while the hash value of x ranges over $\{\pi + r\xi\} = \{\pi + \pi' \mid \pi' \in \Pi'\}$ uniformly. Hence, the information of hash value of x related to the group Π' is independent from the adversary $\mathcal{A}$'s view. □

From Theorem 6.1, we immediately have the following corollary.

Corollary 6.1. *Let* Ξ *be a HPS with diversity, key linearity, and hash linearity. If the derived element* ξ *from diversity is a generator of* Π*, the HPS* Ξ *is perfectly smooth with* $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{Smooth}} = 0$.

Furthermore, we also find that the prefect smoothness implies the diversity as the proof of Theorem 6.1 is reversible. Thus we have that the prefect smoothness is equivalent to the diversity.

Theorem 6.2. *Let* Ξ *be a HPS with key linearity and hash linearity. The HPS* Ξ *is perfectly smooth if and only if* Ξ *is diverse with derived element* ξ *being a generator of* Π.

Proof. We show that the HPS Ξ is diverse if it is perfectly smooth. Fixed on the word $x \in X \setminus L$, the public hash key PK has no information of the hash value of x due to the perfect smoothness. Let SK be the real secret hash key used in hashing x that $\mathsf{PK} = \mathsf{PKGen}(\mathsf{SK})$, which is unknown, and $\pi = \mathsf{Hash}(\mathsf{SK}, x)$. Let ξ be a generator of Π. A random element in Π can be represented as $\pi' = \pi + r\xi$ for some $r \in \mathbb{Z}_{|\Pi|}$. From Eq. (6.2) with the hash linearity, we have that π' is a hash value of x under the key $\mathsf{SK}' = \mathsf{SK} + r\mathsf{SK}^*$ for some $\mathsf{SK}^* \in K$ such that $\mathsf{Hash}(\mathsf{SK}^*, x) = \xi$. From Eq. (6.1) with the key linearity, we have $\mathsf{PKGen}(\mathsf{SK}') = \mathsf{PK} + r\mathsf{PKGen}(\mathsf{SK}^*)$. If $\mathsf{PKGen}(\mathsf{SK}') \neq \mathsf{PK}$, we have that r is fixed and thus π is fixed, which contradicts with the perfect smoothness. Thus we have $\mathsf{PKGen}(\mathsf{SK}') = \mathsf{PK}$ and further $\mathsf{PKGen}(\mathsf{SK}^*) = 0$. Hence, we have that Ξ is diverse with the element ξ such that $\forall x \in X \setminus L, \exists \mathsf{SK}^* \in K, \mathsf{PKGen}(\mathsf{SK}^*) = 0 \wedge \mathsf{Hash}(\mathsf{SK}^*, x) = \xi$. Combined with Corollary 6.1, it completes the proof. □

6.3.2 Generic Constructions of Hash Proof Systems from Diverse Group Systems

In this subsection, we review the Cramer-Shoup constructions of HPS from universal projective hashing derived from diverse group systems [CS02]. We start with the definition of the group system, then the constructions of the universal projective hashing. In the end, we show that the reviewed constructions have *key linearity*, *hash linearity*, and *diversity*. For notational convenience, we use addition for the group operations.

Definition 6.6 (Group System)**.** *Let X, Π be two finite abelian groups, and L be a $\mathcal{NP}$ language with a witness space W and a binary relation R such that $L = \{x \in X \mid \exists w : (x, w) \in R\}$. Let Φ be a finite abelian group of homomorphism $\phi : X \to \Pi$ such that for all $\phi, \phi' \in \Phi$, $x \in X$, and $a \in \mathbb{Z}$, we have $(\phi \pm \phi')(x) = \phi(x) \pm \phi'(x)$ and $(a\phi)(x) = a\phi(x) = \phi(ax)$. If $\phi = 0 \in \Phi$, we have $\phi(x) = 0 \in \Pi$ for all $x \in X$. Let $\mathcal{H}$ be a subgroup of Φ. Then $\mathbf{G} = (\mathcal{H}, X, L, \Pi)$ is a group system.*

Definition 6.7 (Diverse Group System)**.** *A group system $\mathbf{G}$ is diverse if there exists $\phi \in \Phi$ such that $\phi(L) = \langle 0 \rangle$ and $\phi(x) \neq 0$ for all $x \in X \setminus L$.*

Construction 6.1 (Projective Hash Families from Group Systems)**.** *Let $\mathbf{G} = (\mathcal{H}, X, L, \Pi)$ be a group system and $(g_1, \ldots, g_d) \in L$ be a generator of L. A projective hash family $\mathbf{H} = (H, K, X, L, \Pi, S, \alpha)$ defined in [CS02] can be constructed from $\mathbf{G}$ by setting $\{\phi = H_k \mid k \in K\} = \mathcal{H}$ with uniform distribution, $S = \Pi^d$, and $\alpha : K \to S$ that $\alpha(k) = (\phi(g_1), \ldots, \phi(g_d)) = (H_k(g_1), \ldots, H_k(g_d))$. To hash $x \in X$, it simply calculates $H_k(x) = \phi(x) \in \Pi$. If $x \in L$ that $x = \sum_{i=1}^{d} w_i g_i$ where $(w_1, \ldots, w_d) \in W$ is the witnesses of x, it can alternatively calculates $H_k(x) = \phi(\sum_{i=1}^{d} w_i g_i) = \sum_{i=1}^{d} w_i \phi(g_i) \in \Pi$, using $\alpha(k)$ and $(w_1, \ldots, w_d)$.*

Construction 6.2 (HPS from Projective Hash Families)**.** *Let $\mathbf{H} = (H, K, X, L, \Pi, S, \alpha)$ be a projective hash family where L is a $\mathcal{NP}$ language with a witness space W and a binary relation R such that $L = \{x \in X \mid \exists w : (x, w) \in R\}$. A HPS $\Xi = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{PKGen}, \mathsf{Hash}, \mathsf{PHash})$ can be constructed as follows.*

- $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$*: return* $\mathsf{param} = (X, L, W, R, K, S, \Pi)$.
- $\mathsf{SK} \leftarrow \mathsf{SKGen}(\mathsf{param})$*: return* $k \in_R K$.
- $\mathsf{PK} \leftarrow \mathsf{PKGen}(\mathsf{SK})$*: return* $\alpha(k)$.
- $\pi \leftarrow \mathsf{Hash}(\mathsf{SK}, x)$*: return* $H_k(x)$.
- $\pi \leftarrow \mathsf{PHash}(\mathsf{PK}, x, w)$*: return $H_k(x)$ computed using $\alpha(k)$ and a witness w of x without the actual k.*

Let Ξ be a HPS constructed from a group system $\mathbf{G}$ by combining construction 6.1 and construction 6.2. From [CS02], Ξ is *universal* if $\mathbf{G}$ is *diverse*. Furthermore, we need a universal$_2$ HPS, which can be derived from a universal projective hash family.

Construction 6.3 (Universal$_2$ Projective Hash Families)**.** *Let* $\mathbf{H} = (H, K, X, L, \Pi, S, \alpha)$ *be a universal projective hash family,* p *be the smallest prime dividing* $|X \setminus L|$*, and* $\Gamma : X \times E \to \mathbb{Z}_p^n$ *be an injective map. A universal*$_2$ *projective hash family* $\hat{\mathbf{H}} = (\hat{H}, K^{n+1}, X \times E, L \times E, \Pi, S^{n+1}, \hat{\alpha})$ *can be constructed that* $\hat{k} = (k_0, \ldots, k_n) \in K^{n+1}$*,* $\hat{\alpha}(k) = (\alpha(k_0), \ldots, \alpha(k_n)) \in S^{n+1}$*, and* $\hat{H}_{\hat{k}} = H_{k_0}(x) + \langle \Gamma(x, e), (H_{k_1}(x), \ldots, H_{k_n}(x)) \rangle$ *for all* $x \in X$ *and* $e \in E$*.*

Let Ξ_2 be a HPS constructed from a group system $\mathbf{G}$ by combining construction 6.1, construction 6.3, and construction 6.2 in sequence. From [CS02], Ξ_2 is *universal*$_2$ if $\mathbf{G}$ is *diverse*.

Besides the above security properties, we find that HPS from a group system $\mathbf{G} = (\mathcal{H}, X, L, \Pi)$ has some extra properties. Let K be a finite abelian group of order $|\mathcal{H}|$. Since H_k is uniformly distributed over $\mathcal{H}$ by randomly choosing $k \in K$ in construction 6.1, we have that H is a bijection for K and $\mathcal{H}$. Thus we have $H_{k_1} + H_{k_2} = H_{k_1+k_2} \in \mathcal{H}$ for all $k_1, k_2 \in K$.

Theorem 6.3 (Key Linearity)**.** *Let* Ξ *be a universal HPS and* Ξ_2 *be a universal*$_2$ *HPS as constructed above from a group system* $\mathbf{G}$*. The HPSs* Ξ *and* Ξ_2 *have key linearity.*

Proof. As Ξ and Ξ_2 share the same mapping $\alpha : K \to \Pi^d$, we show the linearity of α that for all $k_1, k_2 \in K$,

$$\begin{aligned}
\alpha(k_1) + \alpha(k_2) &= (H_{k_1}(g_1), \ldots, H_{k_1}(g_d)) + (H_{k_2}(g_1), \ldots, H_{k_2}(g_d)) \\
&= (H_{k_1}(g_1) + H_{k_2}(g_1), \ldots, H_{k_1}(g_d) + H_{k_2}(g_d)) \\
&= ((H_{k_1} + H_{k_2})(g_1), \ldots, (H_{k_1} + H_{k_2})(g_d)) \\
&= (H_{k_1+k_2}(g_1), \ldots, H_{k_1+k_2}(g_d)) = \alpha(k_1 + k_2)
\end{aligned}$$

From the linearity of α, we directly have the key linearity of Ξ.

$$\begin{aligned}
&\mathsf{PKGen}(\mathsf{SK}_1) + \mathsf{PKGen}(\mathsf{SK}_2) \\
&= \alpha(\mathsf{SK}_1) + \alpha(\mathsf{SK}_2) = \alpha(\mathsf{SK}_1 + \mathsf{SK}_2) = \mathsf{PKGen}(\mathsf{SK}_1 + \mathsf{SK}_2)
\end{aligned}$$

For Ξ_2, we show the key linearity as follows where $\mathsf{SK}_1 = (k_{1,0}, \ldots, k_{1,n}), \mathsf{SK}_2 =$

$(k_{2,0}, \ldots, k_{2,n}) \in K^{n+1}$.

$$\begin{aligned}
&\mathsf{PKGen}(\mathsf{SK}_1) + \mathsf{PKGen}(\mathsf{SK}_2) \\
&= (a(k_{1,0}), \ldots, a(k_{1,n})) + (a(k_{2,0}), \ldots, a(k_{2,n})) \\
&= (a(k_{1,0}) + a(k_{2,0}), \ldots, a(k_{1,n}) + a(k_{2,n}))) \\
&= (a(k_{1,0} + k_{2,0}), \ldots, a(k_{1,n} + k_{2,n})) = \mathsf{PKGen}(\mathsf{SK}_1 + \mathsf{SK}_2)
\end{aligned}$$

□

Theorem 6.4 (Hash Linearity)**.** *Let Ξ be a universal HPS and Ξ_2 be a universal$_2$ HPS as constructed above from a group system $\mathbf{G}$. The HPSs Ξ and Ξ_2 have hash linearity.*

Proof. Starting from Ξ, we show the hash linearity that

$$\begin{aligned}
&\mathsf{Hash}(SK_1, x) + \mathsf{Hash}(SK_2, x) \\
&= H_{\mathsf{SK}_1}(x) + H_{\mathsf{SK}_2}(x) = H_{\mathsf{SK}_1 + \mathsf{SK}_2}(x) = \mathsf{Hash}(\mathsf{SK}_1 + \mathsf{SK}_2, x)
\end{aligned}$$

Then we show the hash linearity of Ξ_2 as follows where $\mathsf{SK}_1 = (k_{1,0}, \ldots, k_{1,n})$ and $\mathsf{SK}_2 = (k_{2,0}, \ldots, k_{2,n}) \in K^{n+1}$.

$$\begin{aligned}
&\mathsf{Hash}(\mathsf{SK}_1, (x, e)) + \mathsf{Hash}(\mathsf{SK}_2, (x, e)) \\
&= H_{k_{1,0}}(x) + \langle \Gamma(x, e), (H_{k_{1,1}}(x), \ldots, H_{k_{1,n}}(x)) \rangle \\
&\quad + H_{k_{2,0}}(x) + \langle \Gamma(x, e), (H_{k_{2,1}}(x), \ldots, H_{k_{2,n}}(x)) \rangle \\
&= (H_{k_{1,0}}(x) + H_{k_{2,0}}(x)) + \langle \Gamma(x, e), (H_{k_{1,1}}(x) + H_{k_{2,1}}(x), \ldots, H_{k_{1,n}}(x) + H_{k_{2,n}}(x)) \rangle \\
&= H_{k_{1,0}+k_{2,0}}(x) + \langle \Gamma(x, e), (H_{k_{1,1}+k_{2,1}}(x), \ldots, H_{k_{1,n}+k_{2,n}}(x)) \rangle \\
&= \mathsf{Hash}(\mathsf{SK}_1 + \mathsf{SK}_2, (x, e))
\end{aligned}$$

□

Theorem 6.5 (Diversity)**.** *Let Ξ be a universal HPS as constructed above from a group system $\mathbf{G}$. The HPS Ξ is diverse if $|\Pi|$ is prime and $\mathbf{G}$ is diverse.*

Proof. Since $\mathbf{G}$ is diverse, we have that there exists $k \in K$ such that $\mathsf{Hash}(k, x) = 0$ for all $x \in L$ and $\mathsf{Hash}(k, x^*) \neq 0$ for all $x^* \in X \setminus L$. Let $\pi = \mathsf{Hash}(k, x^*) \neq 0$. Since Π is a prime order cyclic group, π is a generator of Π and thus for all $\pi' \in \Pi, \pi' = \mu \cdot \pi$ for some $\mu \in \mathbb{Z}_{|\Pi|}$. By Theorem 6.4, we have that for all $x' \in X \setminus L$,

$$\begin{aligned}
&\pi' = \mathsf{Hash}(k, x') \iff \mu \cdot \pi = \mathsf{Hash}(k, x') \\
&\iff \pi = \mu^{-1} \cdot \mathsf{Hash}(k, x') \iff \pi = \mathsf{Hash}(\mu^{-1} \cdot k, x')
\end{aligned}$$

Hence, for a fixed π, there exists a secret key $\mu^{-1} \cdot k$ that hashes x' to π for all $x' \in X$. Let w be a witness of $x \in L$. Recall the construction of Ξ that $\mathsf{Hash}(k, x) = H_k(x)$ and $\mathsf{PKGen}(k) = \alpha(k) = (H_k(g_1), \ldots, H_k(g_d))$. Since $g_1, \ldots, g_d \in L$ and $H_k(x) = 0$ for all $x \in L$, we have $\mathsf{PKGen}(k) = 0$. Therefore, by Theorem 6.3, we have $\mathsf{PKGen}(\mu^{-1} \cdot k) = \mu^{-1} \cdot \mathsf{PKGen}(k) = 0$ and complete the proof. □

6.4 Generic Construction from Hash Proof Systems

In this section, we describe the key ideas to construct an IND-CCA secure FE-IP scheme. Then we present our FE-IP scheme from HPSs and its security proof.

In a FE-IP scheme, the plaintext vector $\vec{y}$ should be encrypted in a raw form that can be recovered instead of being encrypted as the output of the function $f(\vec{x}, \vec{y})$ so that it can be manipulated by arbitrary $\vec{x}$ to compute $f(\vec{x}, \vec{y})$. In order to achieve the IND-CCA security, the sender who encrypts the message shall provide a Non-Interactive Zero Knowledge (NIZK) proof that it knows the decryption in terms of the raw form of the plaintext vector $\vec{y}$ [RS92]. This is the essential idea of achieving the IND-CCA security. On the other hand, the decryption of a FE involves two parts: the function evaluation and the authorisation to that function evaluation. Obviously, we have to do manipulation first then decryption instead of decryption first then manipulation since the receiver should only be able to compute $f(\vec{x}, \vec{y})$ but not $\vec{y}$ itself. For inner product functionality, the ciphertext of the plaintext vector $\vec{y}$ is manipulated into the ciphertext of $f(\vec{x}, \vec{y}) = \langle \vec{x}, \vec{y} \rangle$ by using the vector $\vec{x}$ and the ciphertext homomorphism. Later, the receiver can decrypt the resulted ciphertext to obtain $f(\vec{x}, \vec{y})$, given the authorisation to $f(\vec{x}, \cdot)$. Before decryption, the receiver also needs to verify the NIZK proof attached to the ciphertext. Using the hash proof system as the NIZK proof system (with auxiliary input), the receiver is required to use the secret key of the HPS to verify the proof. If we consider attacks from outsiders only, it is safe to give the secret key to the end users. However, from the IND-CCA game (Fig. 6.1), we allow attacks from insiders. Therefore, we cannot give the secret key directly to the users. Otherwise, they can generate the proofs without knowing the witnesses. To solve this problem, we limit the scheme to serve at most η users and generate a vector $\vec{\beta}$ of secret keys for the proof system with dimension η. For each user, we randomly pick a vector $\vec{s}$ and compute the inner product $\langle \vec{s}, \vec{\beta} \rangle$ as the secret key for the end user. When encrypting, the sender generates the proof using the individual public keys directly derived for $\vec{\beta}$ as η proof parts. To verify, the receiver assembles the proof parts using $\vec{s}$ and checks with its secret key $\langle \vec{s}, \vec{\beta} \rangle$. Since both $\vec{\beta}$ and $\vec{s}$ have the dimension η, it is impossible to

compute $\vec{\beta}$ with $\eta-1$ pairs of $(\vec{s}, \langle \vec{s}, \vec{\beta} \rangle)$ (i.e. $\vec{\beta}$ is statistically indistinguishable with a random vector). If all η users collude together, they can obtain $\vec{\beta}$ and generate proofs without witnesses but it is meaningless to launch attacks against themselves.

6.4.1 The Construction

Unfortunately, we are not able to construct a generic FE-IP scheme for arbitrary $\mathbb{G}_x$, $\mathbb{G}_y$, and $\mathbb{G}_z$. Due to the definition of hash linearity (Definition 6.4) of HPS, we have to make $\mathbb{G}_x = \mathbb{G}_y = \mathbb{G}_z = \mathbb{Z}_\rho \subset \mathbb{Z}$. To build our FE-IP scheme, we need a diverse HPS with key linearity and hash linearity, a universal$_2$ HPS with hash linearity. Note that the key linearity is used in the proof only. Formally, we present our construction as follows, and recommend the readers to review the related notations defined in Section 2.2.

Construction 6.4 (FE-IP from HPS)**.** *Let* $\Xi_1 = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{PKGen}, \mathsf{Hash}, \mathsf{PHash})$ *be a diverse HPS associated with spaces* (X, L, W, R, K, S, Π) *and* $\Xi_2 = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{PKGen}, \mathsf{Hash}, \mathsf{PHash})$ *be a universal$_2$ HPS associated with space* $(X \times \Pi^\delta, L \times \Pi^\delta, W, R, K', S', \Pi')$ *where* δ *is passed as the input of the algorithm Setup. Both* Ξ_1 *and* Ξ_2 *are required to have the hash linearity.* Ξ_1 *is required to have the key linearity for the security proof. Let* $\xi \in \Pi \setminus \{0\}$ *be an element derived from the diversity of the HPS* Ξ_1*. Our functional encryption scheme for the functionality* $f : \mathbb{Z}_\rho^\delta \times \mathbb{Z}_\rho^\delta \to \mathbb{Z}_\rho$ *works as follows.*

- $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\delta, 1^\eta)$*: Given a security parameter* 1^λ*, a maximum vector size* 1^δ *and a maximum user size* 1^η*, the algorithm generates system-wide parameters* $\mathsf{param}_1 \leftarrow \Xi_1.\mathsf{Setup}(1^\lambda)$ *and* $\mathsf{param}_2 \leftarrow \Xi_2.\mathsf{Setup}(1^\lambda)$*. The algorithm generates two secret key vectors* $\vec{\alpha} = (\alpha_1, \ldots, \alpha_\delta) \in K^\delta$ *and* $\vec{\beta} = (\beta_1, \ldots, \beta_\eta) \in K'^\eta$ *where* $\alpha_i \leftarrow \Xi_1.\mathsf{SKGen}(\mathsf{param}_1)$*,* $\beta_i \leftarrow \Xi_2.\mathsf{SKGen}(\mathsf{param}_2)$*. After that, it generates corresponding public keys* $\vec{A} = (A_1, \ldots, A_\delta) \in S^\delta$ *and* $\vec{B} = (B_1, \ldots, B_\eta) \in S'^\eta$ *where* $A_i = \Xi_1.\mathsf{PKGen}(\alpha_i)$*,* $B_i = \Xi_2.\mathsf{PKGen}(\beta_i)$*. Next, the algorithm packs the public key* $\mathsf{PK} = (\vec{A}, \vec{B})$ *and the master secret key* $\mathsf{MSK} = (\vec{\alpha}, \vec{\beta})$*. Finally, the algorithm publishes* PK *and keeps* MSK *private.*

$$\begin{aligned}
&\mathsf{param}_1 \leftarrow \Xi_1.\mathsf{Setup}(1^\lambda), \quad \mathsf{param}_2 \leftarrow \Xi_2.\mathsf{Setup}(1^\lambda) \\
&\textit{For } i = 1 \ldots \delta, \quad \alpha_i \leftarrow \Xi_1.\mathsf{SKGen}(\mathsf{param}_1), \quad A_i = \Xi_1.\mathsf{PKGen}(\alpha_i) \\
&\textit{For } i = 1 \ldots \eta, \quad \beta_i \leftarrow \Xi_2.\mathsf{SKGen}(\mathsf{param}_2), \quad B_i = \Xi_2.\mathsf{PKGen}(\beta_i)
\end{aligned}$$

  return $(\mathsf{PK}, \mathsf{MSK}) = ((\vec{A}, \vec{B}), (\vec{\alpha}, \vec{\beta}))$*.*

- $\mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, \vec{x})$*: To generate a secret key for the vector* $\vec{x}$*, the algorithm randomly selects a vector* $\vec{s} = (s_1, \ldots, s_\eta) \in_R \mathbb{Z}_\rho^\eta$ *and calculates* K_1 *and* K_2 *as*

follows.

$$\vec{s} \in_R \mathbb{Z}_\rho^\eta, \quad K_1 = \langle \vec{x}, \vec{\alpha} \rangle, \quad K_2 = \langle \vec{s}, \vec{\beta} \rangle.$$

return $\mathsf{SK} = (\vec{x}, \vec{s}, K_1, K_2)$.

– $C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, \vec{y})$*: To encrypt a vector* $\vec{y}$*, the algorithm randomly samples a word* $l \in L$ *with a witness* $w \in W$*. Then the algorithm computes the ciphertext part* $\vec{C} = (C_1, \ldots, C_\delta)$ *where* $C_i = y_i\xi + \Xi_1.\mathsf{PHash}(A_i, l, w)$*. After that, the algorithm computes the proof part* $\vec{\pi} = (\pi_1, \ldots, \pi_\eta)$ *where* $\pi_i = \Xi_2.\mathsf{PHash}(B_i, (l, \vec{C}), w)$*. Finally, the algorithm packs the word, the ciphertext part, and the proof part as one single ciphertext.*

$$\begin{aligned} &(l, w) \in_R R \\ &\textit{For } i = 1 \ldots \delta, \quad C_i = y_i\xi + \Xi_1.\mathsf{PHash}(A_i, l, w) \\ &\textit{For } i = 1 \ldots \eta, \quad \pi_i = \Xi_2.\mathsf{PHash}(B_i, (l, \vec{C}), w) \end{aligned}$$

return $C = (l, \vec{C}, \vec{\pi})$.

– $D \leftarrow \mathsf{Decrypt}(\mathsf{SK}, C)$*: To decrypt, the algorithm assembles the proof parts* $D_2 = \langle \vec{s}, \vec{\pi} \rangle$*. If* $D_2 \neq \Xi_2.\mathsf{Hash}(K_2, (l, \vec{C}))$*, the algorithm outputs* $D = \perp$ *to reject the ciphertext. Otherwise, the algorithm assembles the ciphertext part* $D_1 = \langle \vec{x}, \vec{C} \rangle$*. Then the algorithm decrypts the resulted ciphertext* $D^* = D_1 - \Xi_1.\mathsf{Hash}(K_1, l)$*. Finally, the algorithm extracts the result* $D = \mathrm{sca}_\xi(D^*)$*.*

$$\begin{aligned} D_2 = \langle \vec{s}, \vec{\pi} \rangle, \quad D_2 \stackrel{?}{=} \Xi_2.\mathsf{Hash}(K_2, (l, \vec{C})), \\ D_1 = \langle \vec{x}, \vec{C} \rangle, \quad D^* = D_1 - \Xi_1.\mathsf{Hash}(K_1, l). \end{aligned}$$

return $D = \mathrm{sca}_\xi(D^*)$.

Theorem 6.6. *The construction 6.4 is correct.*

Proof. We verify the correctness by verifying the decryption algorithm.

$$\begin{aligned} D_2 = \langle \vec{s}, \vec{\pi} \rangle &= \sum_{i=1}^{\eta} s_i \pi_i = \sum_{i=1}^{\eta} s_i \Xi_2.\mathsf{PHash}(B_i, (l, \vec{C}), w) = \sum_{i=1}^{\eta} s_i \Xi_2.\mathsf{Hash}(\beta_i, (l, \vec{C})) \\ &= \Xi_2.\mathsf{Hash}(\sum_{i=1}^{\eta} s_i \beta_i, (l, \vec{C})) = \Xi_2.\mathsf{Hash}(\langle \vec{s}, \vec{\beta} \rangle, (l, \vec{C})) = \Xi_2.\mathsf{Hash}(K_2, (l, \vec{C})). \end{aligned}$$

$$D_1 = \langle \vec{x}, \vec{C} \rangle = \sum_{i=1}^{\delta} x_i C_i = \sum_{i=1}^{\delta} x_i \big(y_i \xi + \Xi_1.\mathsf{PHash}(A_i, l, w)\big)$$
$$= \sum_{i=1}^{\delta} x_i y_i \xi + \sum_{i=1}^{\delta} x_i \Xi_1.\mathsf{Hash}(\alpha_i, l) = \langle \vec{x}, \vec{y} \rangle \xi + \Xi_1.\mathsf{Hash}(\sum_{i=1}^{\delta} x_i \alpha_i, l)$$
$$= \langle \vec{x}, \vec{y} \rangle \xi + \Xi_1.\mathsf{Hash}(\langle \vec{x}, \vec{\alpha} \rangle, l) = \langle \vec{x}, \vec{y} \rangle \xi + \Xi_1.\mathsf{Hash}(K_1, l).$$

After verifying D_2 and computing D_1, we compute D as follows and complete the verification.

$$D^* = D_1 - \Xi_1.\mathsf{Hash}(K_1, l) = \langle \vec{x}, \vec{y} \rangle \xi, \quad D = \mathrm{sca}_\xi(D^*) = \mathrm{sca}_\xi(\langle \vec{x}, \vec{y} \rangle \xi) = \langle \vec{x}, \vec{y} \rangle.$$

□

In construction 6.4, we require the calculation of sca_ξ, which may not be computed in polynomial time. If the decryption space $|\{\langle \vec{x}, \vec{y} \rangle\}|$ is polynomial bounded, we can do decryption in an alternative way. In this variation, D_1, D_2, and D^* are computed and checked as normal. To calculate D, we check $D\xi \stackrel{?}{=} D^*$ for each possible $D \in |\{\langle \vec{x}, \vec{y} \rangle\}|$. Since $|\{\langle \vec{x}, \vec{y} \rangle\}|$ is polynomial-sized, the checking algorithm can be done in polynomial time.

In terms of the maximum number η of users, it is not a defect and the IND-CCA model is still suitable for construction 6.4. Since the number of users is polynomial sized, the value of η is also polynomial sized. By observing construction 6.4, we have the size table (Table 6.1).

Table 6.1: Size of Elements in FE-IP (Construction 6.4)

Item	PK	MSK	SK
Size	$\delta\|S\| + \eta\|S'\|$	$\delta\|K\| + \eta\|K'\|$	$(\delta+\eta)\|\mathbb{Z}_\rho\| + \|K\| + \|K'\|$
Item	C		D
Size	$\|L\| + \delta\|\Pi\| + \eta\|\Pi'\|$		$\|\mathbb{Z}_\rho\|$

As long as the value of η is polynomial sized, all the elements in construction 6.4 are polynomial sized. Hence, the limitation on the maximum user number is no longer an issue.

6.4.2 Security Proof

Theorem 6.7. *The proposed FE-IP scheme (construction 6.4), allowing at most η users, is IND-CCA secure (Definition 6.2) if the language (X, L, W, R) associated with both the underlying diverse HPS Ξ_1 and universal$_2$ HPS Ξ_2 satisfies a hard subset membership problem (Definition 2.13).*

Proof. Having a glance at the security proof, we leverage the diversity property of the HPS Ξ_1 to prove the ciphertext indistinguishability of our construction. At the same time, we exploit the universal$_2$ property of the HPS Ξ_2 to finalise the IND-CCA security in terms of dealing the decryption oracle.

In details, we show that an algorithm $\mathcal{S}$ (i.e. simulator) can be constructed to solve SMP in polynomial time with non-negligible probability if an adversary $\mathcal{A}$ can win the IND-CCA game (Fig. 6.1) with non-negligible probability, querying the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}$ at most $\eta - 1$ times and the decryption oracle $\mathcal{O}_{\mathsf{Decrypt}}$ for at most q times. As explained in Section 6.4, it is meaningless to obtain all secret keys of η users and this is the reason why we let the adversary $\mathcal{A}$ query the key generation oracle at most $\eta - 1$ times. Although we limit the maximum number of the key generation queries, we do not limit the maximum number of the decryption queries to a function of η. Therefore, the proof is still in a valid IND-CCA model.

Let $(\Lambda = (X, L, W, R), x^*)$ be an instance of SMP challenged to the simulator $\mathcal{S}$ for distinguishing whether $x^* \in L$ or $x^* \in X \setminus L$ where x^* is sampled from L or $X \setminus L$ with equal probability. To simulate the the IND-CCA game (Fig. 6.1), the simulator $\mathcal{S}$ runs the algorithm Setup as normal to generate a key pair $(\mathsf{PK}, \mathsf{MSK}) = ((\vec{A}, \vec{B}), (\vec{\alpha}, \vec{\beta}))$, and passes the public key PK to the adversary $\mathcal{A}$. Since the simulator $\mathcal{S}$ has the master secret key MSK, it can answer the oracles $\mathcal{O}_{\mathsf{KeyGen}}$ and $\mathcal{O}_{\mathsf{Decrypt}}$ as normal using the master secret key MSK. The adversary $\mathcal{A}$ is restricted to query the secret keys for $\vec{x}$ to $\mathcal{O}_{\mathsf{KeyGen}}$ such that $\langle \vec{x}, \vec{y}_0 \rangle \neq \langle \vec{x}, \vec{y}_1 \rangle$ where $\vec{y}_0$ and $\vec{y}_1$ are the target vectors output by the adversary $\mathcal{A}$ in the next phase. In other words, the adversary $\mathcal{A}$ can only ask the secret keys for $\vec{x}$ such that $\langle \vec{x}, \vec{y}_0 - \vec{y}_1 \rangle = 0$.

At some point, the adversary $\mathcal{A}$ outputs two target vectors $\vec{y}_0$ and $\vec{y}_1$. Then the simulator $\mathcal{S}$ randomly chooses $b \in_R \{0, 1\}$. After that, the simulator $\mathcal{S}$ computes the target ciphertext $C^* = (x^*, \vec{C}^*, \vec{\pi}^*)$ where

$$C_i^* = y_{b,i}\xi + \Xi_1.\mathsf{Hash}(\alpha_i, x^*), \quad \pi_i^* = \Xi_2.\mathsf{Hash}(\beta_i, (x^*, \vec{C}^*)).$$

After receiving the target ciphertext C^*, the adversary $\mathcal{A}$ can continue to query provided oracles as before with the restriction that $\mathcal{A}$ cannot query C^* to the decryption oracle $\mathcal{O}_{\mathsf{Decrypt}}$. Eventually, the adversary $\mathcal{A}$ outputs a bit b'. If $b = b'$, the simulator $\mathcal{S}$ outputs 1, indicating that $\mathcal{A}$ wins the Fig. 6.1. Otherwise, the simulator $\mathcal{S}$ outputs 0. After that, the simulator $\mathcal{S}$ halts in order to complete the simulation.

Let E_L be the event that $\mathcal{S}$ outputs 1 conditioned on $x^* \in L$, and $E_{X \setminus L}$ be the event that $\mathcal{S}$ outputs 1 conditioned on $x^* \in X \setminus L$. Thus we have the advantage

$\mathsf{Adv}_{\mathcal{S}}^{\mathrm{SMP}}$ of solving the subset membership problem.

$$\begin{aligned}\mathsf{Adv}_{\mathcal{S}}^{\mathrm{SMP}} &= |\Pr[1 \leftarrow S \mid x^* \in L] - \Pr[1 \leftarrow S \mid x^* \in X \setminus L]| \\ &= \left|\Pr[E_L] - \Pr[E_{X\setminus L}]\right|\end{aligned} \tag{6.3}$$

For the case of $x^* \in L$, the simulation is perfect since the algorithms ($\Xi_1.\mathsf{PHash}$, $\Xi_2.\mathsf{PHash}$) and ($\Xi_1.\mathsf{Hash}, \Xi_2.\mathsf{Hash}$) are equivalent. Thus we have

$$\left|\Pr[E_L] - \frac{1}{2}\right| = \mathsf{Adv}_{\mathcal{A}}^{\text{IND-CCA}}. \tag{6.4}$$

For the case of $x^* \in X \setminus L$, we modify the game to a new game such that the simulator $\mathcal{S}$ rejects all ciphertexts $C = (l, \vec{C}, \vec{\pi})$ where $l \in X \setminus L$ in the decryption oracle $\mathcal{O}_{\mathsf{Decrypt}}$ in addition to those words, which cannot pass the proof verification. Let E_m be the event that $\mathcal{S}$ outputs 1 conditioned on $x^* \in X \setminus L$ in this modified game, and $E_\perp$ be the event that $l \in X \setminus L$ and $\langle \vec{s}, \vec{\pi} \rangle = \Xi_2.Hash(K_2, (l, \vec{C}))$. In other words, $E_\perp$ is the event that a ciphertext is rejected in the modified game but accepted in the original game. Since the original game and the modified game are identical until event $E_\perp$ occurs, we have

$$\left|\Pr[E_m] - \Pr[E_{X\setminus L}]\right| \leq \Pr[E_\perp]. \tag{6.5}$$

Lemma 6.1. *The event $E_\perp$ occurs in negligible probability as long as Ξ_2 is a universal$_2$ HPS. More precisely, letting $\mathcal{A}$ query $\mathcal{O}_{\mathsf{Decrypt}}$ for at most q times, we have a upper bound of the probability that $E_\perp$ occurs.*

$$\Pr[E_\perp] \leq q \cdot \mathsf{Adv}^{\mathrm{Universal}_2} \tag{6.6}$$

Proof. Let $C = (l, \vec{C}, \vec{\pi})$ be a ciphertext submitted to the decryption oracle $\mathcal{O}_{\mathsf{Decrypt}}$.

- Suppose that $(l, \vec{C}) = (x^*, \vec{C}^*)$, the adversary $\mathcal{A}$ tries to find a $\vec{\pi} \neq \vec{\pi}^*$ such that $\langle \vec{s}, \vec{\pi} \rangle = \langle \vec{s}, \vec{\pi}^* \rangle$. Let $\hat{\vec{\pi}} = \vec{\pi} - \vec{\pi}^* \neq \vec{0}$. Since $\vec{s}$ is independent from $\mathcal{A}$'s view, it is impossible to find a $\hat{\vec{\pi}}$ such that $\langle \vec{s}, \hat{\vec{\pi}} \rangle = 0$.

- Suppose that $(l, \vec{C}) \neq (x^*, \vec{C}^*)$, the adversary $\mathcal{A}$ tries to find a $\vec{\pi}$ such that $\langle \vec{s}, \vec{\pi} \rangle = \Xi_2.\mathsf{Hash}(K_2, (l, \vec{C}))$. Let $\hat{\pi}_i = \Xi_2.\mathsf{Hash}(\beta_i, (l, \vec{C}))$. Since

$$\begin{aligned}\langle \vec{s}, \vec{\pi} \rangle &= \Xi_2.\mathsf{Hash}(K_2, (l, \vec{C})) = \Xi_2.\mathsf{Hash}(\langle \vec{s}, \vec{\beta} \rangle, (l, \vec{C})) \\ &= \sum_{i=1}^{\eta} s_i \Xi_2.\mathsf{Hash}(\beta_i, (l, \vec{C})) = \langle \vec{s}, \hat{\vec{\pi}} \rangle\end{aligned}$$

  and $\vec{s}$ is independent from $\mathcal{A}$'s view, we have $\langle \vec{s}, \vec{\pi} \rangle = \langle \vec{s}, \hat{\vec{\pi}} \rangle \iff \vec{\pi} = \hat{\vec{\pi}}$.

As the scheme allows at most η users (i.e. $\mathcal{A}$ can query $\mathcal{O}_{\mathsf{KeyGen}}$ for at most $\eta - 1$ times), the adversary can get at most $\eta - 1$ pairs of $(\vec{s}, \langle \vec{s}, \vec{\beta} \rangle)$ where all $\vec{s}$ are linearly independent. In the worst case, the vector $\vec{\beta}$ is collapsed into a space of dimension 1 with size $|K'|$ but $\vec{\beta}$ is still uniformly distributed over that space. Let $\hat{\vec{s}} \in \mathbb{Z}_\rho^\eta$ such that $\hat{\vec{s}}$ is linearly independent with all $\vec{s}$ obtained by $\mathcal{A}$. Thus $k = \langle \hat{\vec{s}}, \vec{\beta} \rangle$ is independent from $\mathcal{A}$'s view and uniformly distributed over K'. If $\mathcal{A}$ find a $\vec{\pi}$ such that $\vec{\pi} = \hat{\vec{\pi}}$, we immediately have attacked the universal$_2$ property of Ξ_2 that $\langle \hat{\vec{s}}, \vec{\pi} \rangle = \Xi_2.\mathsf{Hash}(k, (l, \vec{C}))$. This completes the proof of Eq. (6.6).

□

Lemma 6.2. *The hidden bit b is independent from $\mathcal{A}$'s view that*

$$\Pr[E_m] = \frac{1}{2} \tag{6.7}$$

Proof. Thanks to the diversity property of Ξ_1, there exists a $r \in K$ such that $\Xi_1.\mathsf{Hash}(r, x) = \xi$ and $\Xi_1.\mathsf{PKGen}(r) = 0$. Note that we do not need to calculate r. Since $\xi \neq 0$, we have $r \neq 0$. Let $\vec{\gamma} = r \cdot (\vec{y}_b - \vec{y}_{1-b}) \in K^\delta$. Thus we have $\Xi_1.\mathsf{Hash}(\gamma_i, x^*) = (y_{b,i} - y_{1-b,i}) \cdot \Xi_1.\mathsf{Hash}(r, x^*) = y_{b,i}\xi - y_{1-b,i}\xi$. Recall the target ciphertext $\vec{C}^*$ where $C_i^* = y_{b,i}\xi + \Xi_1.\mathsf{Hash}(\alpha_i, x^*)$. Although $\vec{C}^*$ is a ciphertext for $\vec{y}_b$, it can also be a ciphertext for $\vec{y}_{1-b}$ that

$$\begin{aligned} C_i^* &= y_{1-b,i}\xi + \Xi_1.\mathsf{Hash}(\alpha_i + \gamma_i, x^*) \\ &= y_{1-b,i}\xi + \Xi_1.\mathsf{Hash}(\alpha_i, x^*) + \Xi_1.\mathsf{Hash}(\gamma_i, x^*) \\ &= y_{1-b,i}\xi + \Xi_1.\mathsf{Hash}(\alpha_i, x^*) + y_{b,i}\xi - y_{1-b,i}\xi \\ &= y_{b,i}\xi + \Xi_1.\mathsf{Hash}(\alpha_i, x^*). \end{aligned}$$

Thus $\vec{C}^*$ is a ciphertext of $\vec{y}_b$ for $\vec{\alpha}$ or $\vec{y}_{1-b}$ for $\vec{\alpha} + \vec{\gamma}$ where $\vec{\alpha} \neq \vec{\alpha} + \vec{\gamma}$. Since the adversary $\mathcal{A}$ can only request the secret keys for $\vec{x}$ such that $\langle \vec{x}, \vec{y}_0 \rangle = \langle \vec{x}, \vec{y}_1 \rangle$, we have

$$\langle \vec{x}, \vec{\alpha} + \vec{\gamma} \rangle = \langle \vec{x}, \vec{\alpha} \rangle + \langle \vec{x}, \vec{\gamma} \rangle = \langle \vec{x}, \vec{\alpha} \rangle + \langle \vec{x}, r \cdot (\vec{y}_b - \vec{y}_{1-b}) \rangle = \langle \vec{x}, \vec{\alpha} \rangle$$

Hence, the adversary $\mathcal{A}$ cannot distinguish $\vec{\alpha}$ and $\vec{\alpha} + \vec{\gamma}$ from generated keys. Since $\mathsf{PKGen}(\alpha_i + \gamma_i) = \mathsf{PKGen}(\alpha_i) + (y_{b,i} - y_{1-b,i}) \cdot \mathsf{PKGen}(r) = \mathsf{PKGen}(\alpha_i)$, the adversary $\mathcal{A}$ cannot distinguish $\vec{\alpha}$ and $\vec{\alpha} + \vec{\gamma}$ from public keys. Therefore, the hidden bit b is independent from $\mathcal{A}$'s view. □

Combining Eqs. (6.5) to (6.7), we have

$$\left| \Pr[E_{X \setminus L}] - \frac{1}{2} \right| \leq q \cdot \mathsf{Adv}^{\text{Universal}_2}. \tag{6.8}$$

Combining Eqs. (6.3), (6.4) and (6.8), we have

$$\mathsf{Adv}_{\mathcal{A}}^{\text{IND-CCA}} \leq \mathsf{Adv}_{\mathcal{S}}^{\text{SMP}} + q \cdot \mathsf{Adv}^{\text{Universal}_2}. \tag{6.9}$$

From Eq. (6.9), we immediately have the theorem. □

6.5 Instantiations

In this section, we show two instantiations of construction 6.4 from the Decisional Diffie-Hellman (DDH) and Decisional Composite Residuosity (DCR) problems.

6.5.1 From Decisional Diffie-Hellman Assumption

We recall the universal projective hash family proposed by Cramer and Shoup [CS02] derived from a diverse group system based on the DDH problem (Definition 2.14). The key space is $K = \mathbb{Z}_p^2$. For the hash key $k = (s_1, s_2) \in K$, the projection key generation is $\alpha(k) = g_1^{s_1} g_2^{s_2} \in S = \Pi = \mathbb{G}$. To compute the hash value of $x = (X_1, X_2) \in X = \mathbb{G}^2$ with the hash key k, it computes $H_k(x) = X_1^{s_1} X_2^{s_2} \in \Pi$. To compute the have value of $x = (g_1^w, g_2^w) \in L$ with the projection key $\alpha(k)$ and a witness $w \in W = \mathbb{Z}_p$, it computes $H_k(x) = \alpha(k)^w \in \Pi$.

By applying construction 6.2, we obtain a HPS Ξ_1 as follows.

Construction 6.5. *A DDH-based HPS contains the following algorithms.*

- $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$*:* $g_1, g_2 \in_R \mathbb{G}$*, return* $\mathsf{param} = (\mathbb{G}, g_1, g_2)$.
- $\mathsf{SK} \leftarrow \mathsf{SKGen}(\mathsf{param})$*: return* $\mathsf{SK} = (k_1, k_2) \in_R \mathbb{Z}_p^2$.
- $\mathsf{PK} \leftarrow \mathsf{PKGen}(\mathsf{SK})$*: return* $\mathsf{PK} = g_1^{k_1} g_2^{k_2}$.
- $\pi \leftarrow \mathsf{Hash}(\mathsf{SK}, x)$*:* $(x_1, x_2) \leftarrow x$*, return* $\pi = x_1^{k_1} x_2^{k_2}$.
- $\pi \leftarrow \mathsf{PHash}(\mathsf{PK}, x, w)$*: return* $\pi = \mathsf{PK}^w$.

From Theorems 6.3 to 6.5 and $|\Pi| = |\mathbb{Z}_p| = p$, we have that Ξ_1 has key linearity and hash linearity, and is diverse. To ensure that the underlying group system is diverse, we show the existence of ϕ (or equivalent secret key k). Let $r \in \mathbb{Z}_p^+$ and $k = (r, -r \log_{g_2} g_1)$. For all $x = (g_1^w, g_2^w) \in L$, we have $H_k(x) = (g_1^w)^r (g_2^w)^{-r \log_{g_2} g_1} = g_1^0$. For all $x = (g_1^{w_1}, g_2^{w_2}) \in X \setminus L$, we have $H_k(x) = (g_1^{w_1})^r (g_2^{w_2})^{-r \log_{g_2} g_1} = g_1^{r(w_1 - w_2)}$. To simplify our instantiation, we choose $r = 1$ and $x = (g_1^2, g_2)$, and computes $\chi = \Xi_1.\mathsf{Hash}(k, x) = g_1$.

By applying constructions 6.2 and 6.3, we obtain another HPS Ξ_2 as follows, which is universal$_2$.

Construction 6.6. *A DDH-based universal$_2$ HPS contains the following algorithms.*

- $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$*:* $g_1, g_2 \in_R \mathbb{G}$*, return* $\mathsf{param} = (\mathbb{G}, g_1, g_2)$.
- $\mathsf{SK} \leftarrow \mathsf{SKGen}(\mathsf{param})$*: return* $\mathsf{SK} = (k_1, k_2, k_3, k_4) \in_R \mathbb{Z}_p^4$.
- $\mathsf{PK} \leftarrow \mathsf{PKGen}(\mathsf{SK})$*: return* $\mathsf{PK} = (h_1, h_2) = (g_1^{k_1} g_2^{k_2}, g_1^{k_3} g_2^{k_4})$.
- $\pi \leftarrow \mathsf{Hash}(\mathsf{SK}, x, e)$*:* $(x_1, x_2) \leftarrow x$*, return* $\pi = x_1^{k_1} x_2^{k_2} \cdot (x_1^{k_3} x_2^{k_4})^{H(x_1, x_2, e)}$.
- $\pi \leftarrow \mathsf{PHash}(\mathsf{PK}, x, e, w)$*: return* $\pi = (h_1 h_2^{H(x_1, x_2, e)})^w$.

From Theorem 6.4, we have that Ξ_2 has hash linearity. Note that we use a Collision Resistant Hash Function (CRHF) $H : \mathbb{G}^2 \times \mathbb{G}^\delta \to \mathbb{Z}_p$ instead of an injective map Γ when applying construction 6.3.

With Ξ_1 and Ξ_2, we apply construction 6.4 to construct a functional encryption scheme for the inner product functionality $f : \mathbb{Z}_p^\delta \times \mathbb{Z}_p^\delta \to \mathbb{Z}_p$.

Construction 6.7. *Our instantiation works as follows.*

- $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\delta, 1^\eta)$*:*

$$
\begin{aligned}
&H : \mathbb{G}^2 \times \mathbb{G}^\delta \to \mathbb{Z}_p, \quad g_1, g_2 \in_R \mathbb{G}.\\
&\textit{For } i = 1 \ldots \delta, \quad \alpha_i = (\alpha_{i,1}, \alpha_{i,2}) \in_R \mathbb{Z}_p^2, \quad A_i = g_1^{\alpha_{i,1}} g_2^{\alpha_{i,2}}.\\
&\textit{For } i = 1 \ldots \eta, \quad \beta_i = (\beta_{i,1}, \beta_{i,2}, \beta_{i,3}, \beta_{i,4}) \in_R \mathbb{Z}_p^4,\\
&\qquad\qquad\quad B_i = (B_{i,1}, B_{i,2}) = (g_1^{\beta_{i,1}} g_2^{\beta_{i,2}}, g_1^{\beta_{i,3}} g_2^{\beta_{i,4}}).
\end{aligned}
$$

  return $(\mathsf{PK}, \mathsf{MSK}) = ((\vec{A}, \vec{B}), (\vec{\alpha}, \vec{\beta}))$.

- $\mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, \vec{x})$*:*

$$
\begin{aligned}
&\vec{s} \in_R \mathbb{Z}_p^\eta, \quad K_1 = (K_{1,1}, K_{1,2}) = \left(\sum_{i=1}^{\delta} x_i \alpha_{i,1}, \sum_{i=1}^{\delta} x_i \alpha_{i,2} \right)\\
&K_2 = (K_{2,1}, K_{2,2}, K_{2,3}, K_{2,4}) = \left(\sum_{i=1}^{\eta} s_i \beta_{i,1}, \sum_{i=1}^{\eta} s_i \beta_{i,2}, \sum_{i=1}^{\eta} s_i \beta_{i,3}, \sum_{i=1}^{\eta} s_i \beta_{i,4} \right)
\end{aligned}
$$

  return $\mathsf{SK} = (\vec{x}, \vec{s}, K_1, K_2)$.

- $C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, \vec{y})$*:*

$$
\begin{aligned}
&r \in_R \mathbb{Z}_p, \quad l = (u_1, u_2) = (g_1^r, g_2^r), \quad \textit{For } i = 1 \ldots \delta, \quad C_i = g_1^{y_i} \cdot A_i^r\\
&h = H(u_1, u_2, C_1, \ldots, C_\delta), \quad \textit{For } i = 1 \ldots \eta, \quad \pi_i = (B_{i,1} \cdot B_{i,2}^h)^r
\end{aligned}
$$

  return $C = (l, \vec{C}, \vec{\pi})$.

– $D \leftarrow \mathsf{Decrypt}(\mathsf{SK}, C)$*:*

$$h = H(u_1, u_2, C_1, \ldots, C_\delta),$$
$$\prod_{i=1}^{\eta} \pi_i^{s_i} \stackrel{?}{=} u_1^{K_{2,1}+h\cdot K_{2,3}} u_2^{K_{2,2}+h\cdot K_{2,4}}, \quad D^* = \frac{\prod_{i=1}^{\delta} C_i^{x_i}}{u_1^{K_{1,1}} u_2^{K_{1,2}}}$$

return $D = \log_{g_1} D^*$.

As mentioned in Section 6.4, we can decrypt D in an alternative manner instead of calculating $\log_{g_1} D^*$ if $|\{D\}|$ is polynomial-sized.

6.5.2 From Decisional Composite Residuosity Assumption

The DCR problem (Definition 2.15) is to distinguish two probability distributions $\mathcal{D}_P = \{(N, x)\}$ and $\mathcal{D}_{\mathbb{Z}^*_{N^2}\setminus P} = \{(N, x')\}$ where $P = \mathbb{G}_{N'}\mathbb{G}_2\mathbb{G}_T \subset \mathbb{Z}^*_{N^2} = \mathbb{G}_N\mathbb{G}_{N'}\mathbb{G}_2\mathbb{G}_T$. Slightly different from [CS02], we have the corresponding HPS construction as follows with $W = \{0, \ldots, \lfloor N/4 \rfloor\}$, $K = \{0, \ldots, \lfloor N^2/2 \rfloor\}$, $S = L = \mathbb{G}_{N'}$, and $\Pi = X = \mathbb{G}_N\mathbb{G}_{N'}$.

Construction 6.8. *A DCR-based HPS contains the following algorithms.*

– $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$*:* $\mu \in_R \mathbb{Z}^*_{N^2}, g = \mu^{2N} \pmod{N^2}$*, return* $\mathsf{param} = (N, g)$.

– $\mathsf{SK} \leftarrow \mathsf{SKGen}(\mathsf{param})$*: return* $\mathsf{SK} = k \in_R \{0, \ldots, \lfloor N^2/2 \rfloor\}$.

– $\mathsf{PK} \leftarrow \mathsf{PKGen}(\mathsf{SK})$*: return* $\mathsf{PK} = g^k \pmod{N^2}$.

– $\pi \leftarrow \mathsf{Hash}(\mathsf{SK}, x)$*: return* $\pi = x^k \pmod{N^2}$.

– $\pi \leftarrow \mathsf{PHash}(\mathsf{PK}, x = g^w, w)$*: return* $\pi = \mathsf{PK}^w \pmod{N^2}$.

Remark that the value N in Setup is generated as described in Definition 2.15 and g is a generator of $L = \mathbb{G}_{N'}$ with overwhelming probability. As shown in [CS02], the construction 6.5 is derived from a diverse group system where the homomorphism $H_{N'}$ sends all elements in L and no elements in $X \setminus L$ to the group identity 1. More precisely, let g_N be a generator of $\mathbb{G}_N$, and $g_{N'}$ be a generator of $\mathbb{G}_{N'}$. Since N' and N are co-prime and an element $x \in X$ can be represented as $g_N^a g_{N'}^b$ for some $a \in \mathbb{Z}_N$ and $b \in \mathbb{Z}_{N'}$, we have $x^{N'} = (g_N^a g_{N'}^b)^{N'} = g_N^{N'a} g_{N'}^{N'b} = g_N^{aN'} \neq 1$ if $a \neq 0$. Hence, for all $x \in L$ that $a = 0$, we have $x^{N'} = 1$. Again, for all $x \in X \setminus L$ that $a \neq 0$, we have $x^{N'} \neq 1$.

From Theorems 6.3 to 6.5, the HPS Ξ_1 by construction 6.8 has key linearity and hash linearity that can be easily verified as $g^{k_1} g^{k_2} = g^{k_1+k_2}$ and $x^{k_1} x^{k_2} = x^{k_1+k_2}$. However, since $|\Pi|$ is not prime, we have to verify the diversity explicitly. Let $\xi = 1 + N \pmod{N^2}$, which is a generator of $\mathbb{G}_N$ of order N. It is worth noting

that $\xi^a = 1 + aN \pmod{N^2}$ and $\log_\xi x = \frac{x-1 \bmod N^2}{N}$ for all $a \in \mathbb{Z}_N$ and $x \in \mathbb{G}_N$ where the division does not mean the multiplicative inverse but the division of integers. We show that the above HPS has diversity and ξ is a derived element such that $\forall x \in X \setminus L, \exists k \in K, \mathsf{PKGen}(k) = 1 \wedge \mathsf{Hash}(k, x) = \xi$ where 1 is the identity element 0 in Definition 6.5. Let $r = \left(\log_\xi x^{N'}\right)^{-1} \bmod N$ where $x^{N'} \in \mathbb{G}_N$ and the multiplicative inverse is computable for all $x \in \mathbb{G}_N\mathbb{G}_{N'}$ with overwhelming probability. Let $k = rN'$. Since g is a generator of $\mathbb{G}_{N'}$ of order N', we have $\mathsf{PKGen}(k) = g^k = g^{rN'} = 1 \pmod{N^2}$. Since $r^{-1} = \log_\xi x^{N'} \iff x^{N'} = \xi^{r^{-1}}$, we have $\mathsf{Hash}(k, x) = x^{rN'} = (x^{N'})^r = (\xi^{r^{-1}})^r = \xi$. Hence, the diversity is verified with the derived element $\xi = 1 + N \pmod{N^2}$.

Similarly to the DDH instantiation, by applying constructions 6.2 and 6.3, we obtain a universal$_2$ HPS Ξ_2 with key linearity and hash linearity as follows.

Construction 6.9. *A DCR-based universal$_2$ HPS contains the following algorithms.*

- $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$*:* $\mu \in_R \mathbb{Z}^*_{N^2}, g = \mu^{2N} \pmod{N^2}$*, return* $\mathsf{param} = (N, g)$.
- $\mathsf{SK} \leftarrow \mathsf{SKGen}(\mathsf{param})$*: return* $\mathsf{SK} = (k_1, k_2) \in_R \{0, \ldots, \lfloor N^2/2 \rfloor\}^2$.
- $\mathsf{PK} \leftarrow \mathsf{PKGen}(\mathsf{SK})$*: return* $\mathsf{PK} = (h_1, h_2) = (g^{k_1}, g^{k_2}) \pmod{N^2}$.
- $\pi \leftarrow \mathsf{Hash}(\mathsf{SK}, x, e)$*: return* $\pi = x^{k_1} \cdot x^{k_2 \cdot H(x,e)} \pmod{N^2}$.
- $\pi \leftarrow \mathsf{PHash}(\mathsf{PK}, x = g^w, e, w)$*: return* $\pi = (h_1 h_2^{H(x,e)})^w \pmod{N^2}$.

With Ξ_1 and Ξ_2, we apply construction 6.4 to construct a functional encryption scheme for the inner product functionality[1] $f : \mathbb{Z}_N^\delta \times \mathbb{Z}_N^\delta \to \mathbb{Z}_N$. Let $\xi = 1 + N \pmod{N^2}$.

Construction 6.10. *Our instantiation works as follows.*

- $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\delta, 1^\eta)$*:*

$$
\begin{aligned}
&H : \mathbb{Z}_{N^2} \times \mathbb{Z}_{N^2}^\delta \to \{0, \ldots, 2^\lambda - 1\}, \quad \mu \in_R \mathbb{Z}^*_{N^2}, \quad g = \mu^{2N} \pmod{N^2}.\\
&\textit{For } i = 1 \ldots \delta, \quad \alpha_i \in_R \{0, \ldots, \lfloor N^2/2 \rfloor\}, \quad A_i = g^{\alpha_i} \pmod{N^2}.\\
&\textit{For } i = 1 \ldots \eta, \quad \beta_i = (\beta_{i,1}, \beta_{i,2}) \in_R \{0, \ldots, \lfloor N^2/2 \rfloor\}^2,\\
&\qquad\qquad\quad B_i = (B_{i,1}, B_{i,2}) = (g^{\beta_{i,1}}, g^{\beta_{i,2}}) \pmod{N^2}.
\end{aligned}
$$

  return $(\mathsf{PK}, \mathsf{MSK}) = ((\vec{A}, \vec{B}), (\vec{\alpha}, \vec{\beta}))$.

- $\mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, \vec{x})$*: The elements K_1 and K_2 are computed over $\mathbb{Z}$.*

$$
\vec{s} \in_R \mathbb{Z}_N^\eta, \quad K_1 = \sum_{i=1}^{\delta} x_i \alpha_i, \quad K_2 = (K_{2,1}, K_{2,2}) = \left(\sum_{i=1}^{\eta} s_i \beta_{i,1}, \sum_{i=1}^{\eta} s_i \beta_{i,2} \right).
$$

[1] We do not fully use the key space (i.e. $|K| = \lfloor N^2/2 \rfloor > N$).

return $\mathsf{SK} = (\vec{x}, \vec{s}, K_1, K_2)$.

– $C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, \vec{y})$*:*

$$w \in_R \{0, \ldots, \lfloor N/4 \rfloor\}, \quad l = g^w \pmod{N^2},$$
$$\textit{For } i = 1 \ldots \delta, \quad C_i = \xi^{y_i} \cdot A_i^w \pmod{N^2}$$
$$h = H(l, C_1, \ldots, C_\delta), \quad \textit{For } i = 1 \ldots \eta, \quad \pi_i = (B_{i,1} \cdot B_{i,2}^h)^w$$

return $C = (l, \vec{C}, \vec{\pi})$.

– $D \leftarrow \mathsf{Decrypt}(\mathsf{SK}, C)$*:*

$$h = H(l, C_1, \ldots, C_\delta), \quad \prod_{i=1}^{\eta} \pi_i^{s_i} \stackrel{?}{=} l^{K_{2,1} + h \cdot K_{2,2}} \pmod{N^2},$$
$$D^* = \frac{\prod_{i=1}^{\delta} C_i^{x_i}}{l^{K_1}} \pmod{N^2}, \quad D = \frac{D^* - 1 \mod N^2}{N}.$$

return D.

6.6 Chapter Summary

In this chapter, we defined new properties of Hash Proof System (HPS). We found that the existing HPS constructions by [CS02] have those new properties. As the main contribution of this chapter, we proposed an IND-CCA secure Functional Encryption for Inner Products scheme, which can be generically constructed from a diverse HPS with key linearity and hash linearity, and a universal$_2$ HPS with hash linearity. Moreover, we constructed two concrete schemes from DDH and DCR assumptions via our proposed generic construction.

One of our future work will be relaxing the scheme without limiting the number of user who can decrypt. Another future work will be finding new HPS, which has our defined properties, from other SMPs so that we can construct new FE schemes under new assumptions.

Chapter 7

Hierarchical Functional Encryption for Linear Transformations

In contrast to the conventional all-or-nothing encryption, Functional Encryption (FE) allows partial revelation of encrypted information based on the keys associated with different functionalities. Extending FE with key delegation ability, Hierarchical Functional Encryption (HFE) enables a secret key holder to delegate a portion of its decryption ability to others and the delegation can be done hierarchically. All HFE schemes in the literature are for general functionalities and not very practical. In this chapter, we focus on the functionality of linear transformations (i.e. matrix product evaluation). We refine the definition of HFE and further extend the delegation to accept multiple keys. We also propose a generic HFE construction for linear transformations with Indistinguishability under Chosen Plaintext Attacks (IND-CPA) security in the standard model from Hash Proof System (HPS). In addition, we give two instantiations from the Decisional Diffie-Hellman (DDH) and Decisional Composite Residuosity (DCR) assumptions which to the best of our knowledge are the first practical concrete HFE constructions.

7.1 Introduction

Encryption can provide confidentiality and privacy for our sensitive information in a variety of ways. Typically, conventional encryption is in an all-or-nothing fashion that an entity is able to either access all the encrypted information or nothing, excepting the message length. In contrast, Functional Encryption (FE) [BSW11] allows partial revelation of encrypted information based on the keys associated with different functionalities. Precisely, Alice can encrypt some message m under Bob's public key, and send the ciphertext to a public domain where Charlie can access. Later, Bob issues a secret key for a function f to Charlie using a master secret key corresponding to Bob's public key. As a result, Charlie is able to learn $f(m)$ from the ciphertext and the secret key received from Bob but nothing else. Usually, the function f is a universal function for a function class $\mathcal{F}$ indexed by a function key k from a key space K s.t. $\mathcal{F} = \{f_k \mid k \in K\}$. In this case, Bob sends a secret key

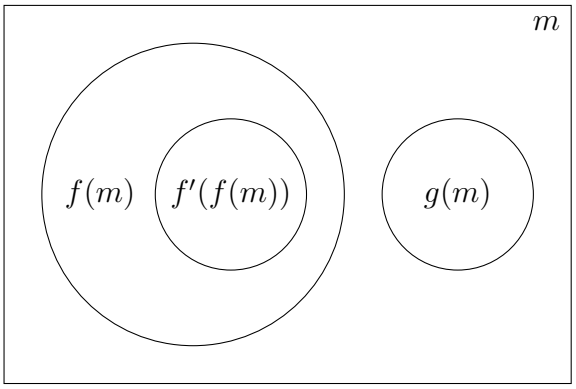

Figure 7.1: Nested Information Hierarchy

for a function key k to Charlie who learns $f(k, m)$ or simply $f_k(m)$.

When Bob can control the amount of the information that Charlie can reveal, it leads to a natural question of whether Charlie is able to further let other people (e.g. David) obtain a narrower portion of his decryption ability. Hierarchical Functional Encryption (HFE) [ABG+13, BS15, CGJS15] gives an affirmative answer. In HFE, Charlie with a secret key associated with a function f is able to generate a new secret key associated with a composed function $f' \circ f$ for David without the help of Bob. Upon receiving the key from Charlie, David can learn $f'(f(m))$ but nothing else of the original message m, as illustrated in the information hierarchy in Fig. 7.1. It is worth noting that the above delegation process is repeatable that David can delegate a portion of his decryption ability to other people, making the hierarchy grow deeper and deeper.

In this chapter, we focus on HFE for the functionality of matrix product evaluation, which implies the function class of linear transformation via transformation matrices. More precisely, the functionality is defined as $f(\mathbf{A}, \mathbf{X}) = f_{\mathbf{A}}(\mathbf{X}) = \mathbf{AX}$ where the matrix $\mathbf{A}$ is the transformation matrix (i.e. the function key) and the matrix $\mathbf{X}$ is the message to be encrypted. It is easy to see that such a functionality is a natural generalisation of the functionality of inner product evaluation [ABCP16, ABDCP15, ALS16, ZMY17].

There are a few practical applications of HFE for the functionality of matrix product evaluation, including but not limited to descriptive statistics. Differing from the functional encryption for inner product evaluation [ABCP16, ABDCP15, ALS16, ZMY17], our proposed HFE allows the evaluation to be done in a levelled manner. We take the calculation of the weighted sum as an example. Suppose Alice is a researcher in a consulting firm conducting marketing research on the demand of various products and the company will sell Alice's results to different clients. Once the research results are ready, Alice encrypts the demand level of different areas for each product (e.g. $\mathbf{X} = \begin{bmatrix} 2 & 1 & 9 & 0 & 6 & 2 & 5 & 6 & 1 \end{bmatrix}^\top$ for the demand

of coffee in different regions) under her company's public key. As the manager of Alice's company, Bob has the master secret key and decides to sell Alice's result in different levels and at different prices. Suppose Charlie wants to buy from Bob some information he is interested in. He does not want all the details but a overall score for the demand of coffee, and he is more interested in the areas near his store, i.e. those areas will have a higher weight. Hence, Charlie buys a secret key of a weighted matrix (e.g. $\mathbf{A} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 3 & 2 & 1 & 0 \end{bmatrix}$) from Bob for a low price so that he can learn a summary of the market demand (e.g. $\mathbf{AX} = 65$ for coffee).

With HFE, Bob is also able to sell Alice's result to some regional resellers (or proxies), e.g. David. Unlike Charlie, David wants the details of some regions and a summary for other regions. Thus he buys a secret key of a weighted matrix (e.g. $\mathbf{A} = \left[\begin{array}{c|cccc} \mathbf{I}_5 & \multicolumn{4}{c}{\mathbf{0}_{5,4}} \\ \hline \mathbf{0}_{1,5} & 5 & 4 & 1 & 2 \end{array}\right]$) from Bob for a higher price so that he learns Alice's partial result (e.g. $\mathbf{AX} = \begin{bmatrix} 2 & 1 & 9 & 0 & 6 & 38 \end{bmatrix}^\top$ for coffee). Later, David can resell what he obtained from Bob to other clients in his region. For example, another store manager Eve in David's region is interested in Alice's result, and decides to buy some market information for her nearby regions from David. In the reselling process, David uses his secret key for $\mathbf{A}$ to generate a new secret key for $\mathbf{C} = \mathbf{BA}$ based on Eve's demand (e.g. $\mathbf{B} = \begin{bmatrix} 20 & 15 & 10 & 5 & 2 & 0 \end{bmatrix}$ and thus $\mathbf{C} = \begin{bmatrix} 20 & 15 & 10 & 5 & 2 & 0 & 0 & 0 & 0 \end{bmatrix}$). Once Eve gets the key, she can learn $\mathbf{CX}$ from Alice's research (e.g. $\mathbf{CX} = 157$ for coffee). On the other hand, David can also resell his data obtained from Bob to other resellers in different levels and at different prices.

There are many potential applications of HFE for linear transformation, such as those related to descriptive statistics as demonstrated above. In this chapter, we introduce and formalise this useful cryptographic primitive and present a generic construction of it. We also show two practical instantiations of the generic construction based on some standard assumptions.

7.1.1 Related Work

The formal study of Functional Encryption (FE) was initiated by Boneh et al. [BSW11] with syntax and security model defined for general functionality. In this chapter, we focus on the public-key FE. Other than Predicate Encryption (PE) [KSW08] (a subclass of FE) and theoretical constructions [GGH+13] for arbitrary functionality, Abdalla et al. [ABDCP15] focused on the functionality of inner product evaluation that only the inner product $\langle \vec{x}, \vec{y} \rangle$ of the encrypted vector $\vec{y}$ is revealed with a secret key for the vector $\vec{x}$. Furthermore, the authors challenged to construct practical schemes with such a functionality, and proposed an s-IND-CPA secure

generic construction from any s-IND-CPA Public Key Encryption (PKE) schemes with the properties of randomness reuse, linear key homomorphism, and linear ciphertext homomorphism under shared randomness. Precisely, Selective Security against Chosen Plaintext Attacks (s-IND-CPA) is a weaker notion of the standard Indistinguishability under Chosen Plaintext Attacks (IND-CPA) where the adversary is required to submit the target plaintexts before receiving the public key from the challenger. Based on [ABDCP15], Abdalla et al. [ABCP16] enhanced their previous generic construction [ABDCP15] to the standard IND-CPA security from any s-IND-CPA PKE schemes with the properties of linear key homomorphism, linear ciphertext homomorphism under shared randomness, ℓ-public-key reproducibility, and ℓ-ciphertext reproducibility. The authors also showed several instantiations from various s-IND-CPA PKE schemes based on Decisional Diffie-Hellman (DDH) assumption [DH76], Decisional Composite Residuosity (DCR) assumption [BCP03, Pai99], and Learning With Error (LEW) assumption [Reg05]. Independent from [ABCP16], Agrawal et al. [ALS16] constructed FE schemes for inner products directly from DDH, DCR, and LEW assumptions instead of a generic construction, and obtained better efficiency. It is worth noting that the proofs of the DDH construction and the DCR construction in [ALS16] are implicitly built on Hash Proof System (HPS) [CS02] as the HPS makes the secret keys simulatable and IND-CPA security achievable. In contrast to [ALS16], Zhang et al. [ZMY17] (see Chapter 6 for details) recently provided another framework of constructing Functional Encryption for Inner Products (FE-IP) explicitly from HPS with the properties of key linearity, hash linearity and diversity. Also, the authors further improved the security from IND-CPA to Indistinguishability under adaptive Chosen Ciphertext Attacks (IND-CCA).

There are several extension works [ARW16, ABG+13, BJK15] on Functional Encryption. As one of those extensions, Hierarchical Functional Encryption (HFE) enables delegation capability, which is initially mentioned in [ABG+13], generalising the notions of hierarchical identity-based encryption [BBG05] and hierarchical predicate encryption [LOS+10] for more expressive access controls. In particular, [ABG+13, BS15, CGJS15] define HFE as a normal FE with an extra delegation algorithm that takes a function key SK_f and a function f' and outputs a function key $\mathsf{SK}_{f' \circ f}$. As mentioned before (Fig. 7.1), the newly generated key $\mathsf{SK}_{f' \circ f}$ allows revelation of $f'(f(m))$ but nothing else from the encrypted message m. In the literature, Ananth et al. [ABG+13] and Chandran et al. [CGJS15] purposed general purpose HFE constructions direct from indistinguishable obfuscation ($i\mathcal{O}$) with fixed depth where the delegation process can only be proceeded for a fixed number of times. As an improvement, Brakerski et al. [BS15] proposed a generic transformation from any general purpose FE to a general purpose HFE with unbounded depth. The

transformation requires a private-key encryption scheme and a puncturable pseudorandom function family, and does not rely on $i\mathcal{O}$. As far as we know, all the HFE schemes in the literature are general purposed and not very practical.

7.1.2 Our Contribution

In this chapter, we refine and simplify the definition of Hierarchical Functional Encryption (HFE) originally from [ABG+13, BS15, CGJS15]. We merge the delegation algorithm $\mathsf{SK}_{f' \circ f} \leftarrow \mathsf{Delegate}(\mathsf{SK}_f, f')$ with the key generation algorithm $\mathsf{SK}_{f_k} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, k)$ to form a new key generation algorithm $\mathsf{SK}_{f' \circ f} \leftarrow \mathsf{KeyGen}(\mathsf{SK}_f, f')$, making the master secret key equivalent to a secret key of an identity function $\mathsf{MSK} \stackrel{\text{def}}{=} \mathsf{SK}_{\mathsf{id}}$. As a result, our definition of HFE consists of four algorithms instead of five algorithms, and it is compatible with the IND-CPA security model defined for the normal FE, which is much simpler than the previously defined model for HFE. In terms of the evolution of encryption, our definition of HFE is a more natural generalisation of Hierarchical Identity-Based Encryption (HIBE) [BBG05].

As an extension of HFE, we introduce the notion of extended Hierarchical Functional Encryption (eHFE), allowing delegation from multiple secret keys. Precisely, in eHFE, the key generation algorithm takes n secret keys $\mathsf{SK}_{f_1}, \ldots, \mathsf{SK}_{f_n}$ for functions $f_1, \ldots, f_n$ correspondingly and a delegation function f', and it generates a new secret key SK_f for a function f such that $f(x) = f'(f_1(x), \ldots, f_n(x))$.

We derive the definition of Hierarchical Functional Encryption for Linear Transformations (HFE-LT) from HFE (similarly, extended Hierarchical Functional Encryption for Linear Transformations (eHFE-LT) from eHFE) for the functionality of matrix product evaluation. Different from the previous general results which are of theoretical interest, we propose a generic and practical construction of HFE-LT. It is worth noting that our construction has unbounded depth in delegation (Table 7.1). Since the HFE-LT scheme cannot be trivially constructed from function encryption for inner products, our generic construction is explicitly built from HPS with key linearity, hash linearity and diversity, and achieves IND-CPA security in the standard model. As an extension, we also adapt our generic HFE-LT construction to a generic eHFE-LT construction. We provide two practical HFE-LT instantiations based on the DDH assumption and the DCR assumption. In the DDH instantiation, the decryption space is limited to be of polynomial size so that the decryption can be done in polynomial time. However, in the DCR instantiation, there is no such limitation. In addition, both instantiations can be extended to the eHFE-LT setting.

Table 7.1: Comparison of Different FE Schemes

Scheme	Functionality	Assumption	Hierarchical	Depth	Multi-Key
[ABG+13]	Generic	$i\mathcal{O}$	Yes	Constant	No
[CGJS15]	Generic	$i\mathcal{O}$ + HIBE	Yes	Fixed	No
[BS15]	Generic	Generic FE	Yes	Unbounded	No
[ABCP16]	Inner Product	s-IND-CPA PKE: DDH/DCR/LWE	No	$\times$	No
[ALS16]	Inner Product	DDH/DCR/LWE	No	$\times$	No
Our HFE-LT	Matrix Product	Diverse HPS: DDH/DCR	Yes	Unbounded	No
Our eHFE-LT	Matrix Product	Diverse HPS	Yes	Unbounded	Yes

7.1.3 Chapter Organisation

The rest of this chapter is organised as follows. It is strongly suggested to read the preliminaries in Chapter 2, especially the notations used in this thesis, the definition of FE in Chapter 3, and the extended HPS in Section 6.3. As our main contribution, we refine HFE and define HFE-LT in Section 7.2. The generic construction from HPS is proposed in Section 7.3 with the security proof. In Section 7.4, we adapt HFE to eHFE in terms of the definition, the generic construction, and the security proof. After that, we propose our concrete HFE-LT constructions instantiated from DDH and DCR assumptions in Section 7.5. Finally, this chapter is summarised in Section 7.6.

7.2 Formal Definitions

In this section, we briefly review the definition of Hierarchical Functional Encryption (HFE) in the literature. After that, we formalise our refined definition of HFE.

In [ABG+13, BS15, CGJS15], the HFE is defined as a normal FE with an extra delegation algorithm $\mathsf{Delegate}$. The (probably) randomised algorithm $\mathsf{SK}_{f' \circ f} \leftarrow \mathsf{Delegate}(\mathsf{SK}_f, f')$ takes a secret key SK_f for a function $f : X \to Z \subset Y$, and an another function $f' : Z \to Z' \subset Y$ as inputs where Z, Z' are the images of the functions f, f' correspondingly. It generates a new secret key $\mathsf{SK}_{f' \circ f}$ for the composed function $f' \circ f : X \to Z' \subset Y$. By observing the key generation algorithm $\mathsf{SK}_{f_k} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, k)$ and the delegation algorithm $\mathsf{SK}_{f' \circ f} \leftarrow \mathsf{Delegate}(\mathsf{SK}_f, f')$, we find that the algorithm KeyGen is actually a special case of the algorithm $\mathsf{Delegate}$ when the master secret key MSK is considered as a secret key $\mathsf{SK}_{\mathsf{id}}$ of an identity function. More precisely, we have $\mathsf{KeyGen}(\mathsf{MSK}, k) \stackrel{\text{def}}{=} \mathsf{Delegate}(\mathsf{SK}_{\mathsf{id}}, f_k) \to \mathsf{SK}_{f_k \circ \mathsf{id}} = \mathsf{SK}_{f_k}$ with $\mathsf{MSK} \stackrel{\text{def}}{=} \mathsf{SK}_{\mathsf{id}}$. Therefore, the duplicated algorithm can be removed, and it becomes a more natural generalisation of Hierarchical Identity-Based Encryption (HIBE) [BBG05] as there is no $\mathsf{Delegate}$ algorithm in HIBE. The refined HFE is formalised as follows.

$$
\begin{array}{l||l}
\underline{\mathsf{Game}^{\lambda}_{\text{IND-CPA}}} & \underline{\mathcal{O}_{\mathsf{KeyGen}}(k)} \\
\quad (\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^{\lambda}) & \quad \mathcal{K} \leftarrow \mathcal{K} \cup \{k\} \\
\quad (x_0, x_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}}}(\mathsf{PK}) & \quad \text{return } \mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, k) \\
\quad b \in_R \{0,1\} & \underline{\mathcal{O}_{\mathsf{KeyGen}}(f, f')} \\
\quad C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, x_b) & \quad k \leftarrow K \text{ s.t. } f_k = f' \circ f \in \mathcal{F} \\
\quad b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}}}(C) & \quad \text{return } \mathsf{SK} \leftarrow \mathcal{O}_{\mathsf{KeyGen}}(k)
\end{array}
$$

$$
\mathsf{Adv}^{\text{IND-CPA}}_{\mathcal{A}} = \left| \Pr\left[b = b' \,|\, \forall k \in \mathcal{K},\ f_k(x_0) = f_k(x_1) \right] - \frac{1}{2} \right|
$$

Figure 7.2: IND-CPA Game resolved for HFE

Definition 7.1 (Hierarchical Functional Encryption)**.** *A Hierarchical Functional Encryption (HFE) for a function class $\mathcal{F} = \{f_k : X \to Y \mid k \in K\}$ consists of the following four polynomial time algorithms:*

$$
\begin{aligned}
(\mathsf{PK}, \mathsf{MSK}) &\leftarrow \mathsf{Setup}(1^{\lambda}), & \mathsf{SK}_{f' \circ f} &\leftarrow \mathsf{KeyGen}(\mathsf{SK}_f, f'), \\
C &\leftarrow \mathsf{Encrypt}(\mathsf{PK}, x), & y = f(x) &\leftarrow \mathsf{Decrypt}(\mathsf{SK}_f, C).
\end{aligned}
$$

While other three algorithms work as in Definition 3.1, the refined key generation algorithm KeyGen *takes a secret key for function $f \in \mathcal{F} : X \to Z \subset Y$, and a function $f' : Z \to Z' \subset Y$ as inputs such that $f_k = f' \circ f \in \mathcal{F}$ for some $k \in K$. The algorithm* KeyGen *outputs a secret key for the function f_k. The master secret key is defined as a secret key for the identity function (*$\mathsf{MSK} \stackrel{\text{def}}{=} \mathsf{SK}_{\mathsf{id}}$*) and thus it can be directly used in the decryption algorithm* Decrypt*.*

Remark 7.1. *The secret key holders should be careful in the secret key generation. If the delegation function f' is invertible, the newly generated secret key $\mathsf{SK}_{f' \circ f} \leftarrow \mathsf{KeyGen}(\mathsf{SK}_f, f')$ is equivalent to the original secret key SK_f since*

$$
\mathsf{KeyGen}(\mathsf{SK}_{f' \circ f}, f'^{-1}) = \mathsf{SK}_{f'^{-1} \circ f' \circ f} = \mathsf{SK}_{\mathsf{id} \circ f} = \mathsf{SK}_f.
$$

As the above operations can be done in the ideal world, it is not considered as a security issue in the real world, even with a secure scheme.

Since the syntax is similar to the normal FE, the definition of the IND-CPA security (Definition 3.2) can be re-used for HFE. When the adversary $\mathcal{A}$ query the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}$ for functions f and f' to obtain a new secret key $\mathsf{SK}_{f' \circ f}$ in HFE, it can be resolved by querying the original oracle $\mathcal{O}_{\mathsf{KeyGen}}$ with a function key k since $f_k = f' \circ f \in \mathcal{F}$ for some $k \in K$. The resolved security model is illustrated in Fig. 7.2. Using the security model of FE for HFE, we implicitly require that the secret key and the delegated key have the same distribution.

With the refined definition of HFE, we derive the syntax of our HFE for Linear

Transformations.

Definition 7.2 (Hierarchical Functional Encryption for Linear Transformations)**.** *Let $\mathbb{R}$ be a ring, $K = \{\mathbf{A} \mid \mathbf{A} \in \mathbb{R}^{i\times\delta}, i \in \mathbb{Z}^+\}$, $X = \mathbb{R}^{\delta\times\gamma}$, and $Y = \{\mathbf{Y} \mid \mathbf{Y} \in \mathbb{R}^{i\times\gamma}, i \in \mathbb{Z}^+\}$. The universal function $f : K \times X \to Y$ is defined as $f_{\mathbf{A}} : \mathbf{X} \mapsto \mathbf{AX}$. In short, the transformation function $f_{\mathbf{A}}$ is simply denoted by the internal transformation matrix $\mathbf{A}$. A Hierarchical Functional Encryption for Linear Transformations (HFE-LT) is an HFE for a function class $\mathcal{F} = \{f_k : X \to Y \mid k \in K\}$, consisting of the following four polynomial time algorithms:*

$$(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\delta, 1^\gamma), \qquad \mathsf{SK}_{\mathbf{BA}} \leftarrow \mathsf{KeyGen}(\mathsf{SK}_{\mathbf{A}}, \mathbf{B}),$$
$$C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, \mathbf{X}), \qquad \mathbf{Y} = \mathbf{AX} \leftarrow \mathsf{Decrypt}(\mathsf{SK}_{\mathbf{A}}, C).$$

It is clear that the system setup algorithm Setup specifies the dimensions of K, X, and Y by the extra inputs δ and γ. Again, the master secret key is a secret key for the identity matrix that $\mathsf{MSK} \stackrel{\text{def}}{=} \mathsf{SK}_{\mathbf{I}}$. Since the properties of ring are not fully used in matrix multiplication, the above definition can be extended to any algebraic structure (even different structures for K, X, and Y) as long as $(\mathbf{BA})\mathbf{X} = \mathbf{B}(\mathbf{AX})$ with all operations valid. In addition to the key generation algorithm KeyGen, if the delegation matrix $\mathbf{B}$ is invertible, the newly generated key is equivalent to the original key since $\mathsf{KeyGen}(\mathsf{KeyGen}(\mathsf{SK}_{\mathbf{A}}, \mathbf{B}), \mathbf{B}^{-1}) = \mathsf{SK}_{\mathbf{B}^{-1}\mathbf{BA}} = \mathsf{SK}_{\mathbf{A}}$.

7.3 Generic Construction from Hash Proof Systems

In this section, we propose a generic construction of HFE-LT from HPS and prove its security. It is strongly recommended that the readers should read Section 2.2 in advance since our construction is based on the notations defined in Section 2.2. We also suggest that the readers should review the extended HPS described in Section 6.3.

7.3.1 The Construction

Let $\Xi = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{PKGen}, \mathsf{Hash}, \mathsf{PHash})$ be a diverse HPS associated with an Subset Membership Problem (SMP) instance $\Lambda = (X, L, W, R)$ and further spaces (K, S, Π). The HPS Ξ is required to have hash linearity for completeness and key linearity for soundness. Let $\xi \in \Pi \setminus \{0\}$ be an element derived from the diversity of the HPS Ξ, and n be the order of the group $\Pi' = \{a\xi \mid a \in \mathbb{Z}\}$. Our HFE-LT with

$\mathbb{R} = \mathbb{Z}_n$ works as follows[1].

– $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\delta, 1^\gamma)$: Given a security parameter 1^λ and the size $\delta \times \gamma$ of message matrices, the system setup algorithm generates a system-wide parameter $\mathsf{param} \leftarrow \Xi.\mathsf{Setup}(1^\lambda)$. The algorithm generates a key matrix $\mathbf{K_I}$ of size $\delta \times \gamma$ as the core part of the secret key of the identity matrix $\mathbf{I}_\delta \in \mathbb{Z}_n^{\delta \times \delta}$ by invoking $\Xi.\mathsf{SKGen}(\mathsf{param})$ for $\delta \times \gamma$ times. After that, the algorithm generates the corresponding public keys $\mathbf{P} = \Xi.\mathsf{PKGen}(\mathbf{K_I})$. The full secret key for the identity matrix is packed as $\mathsf{SK_I} = (\mathbf{I}_\delta, \mathbf{K_I})$. Finally, the algorithm publishes $\mathbf{P}$ as the public key PK, and keeps $\mathsf{SK_I}$ as the master secret key MSK.

$$\begin{aligned}
&\mathsf{param} \leftarrow \Xi.\mathsf{Setup}(1^\lambda),\\
&\mathbf{K_I} = \begin{bmatrix} k_{1,1} & \cdots & k_{1,\gamma} \\ \vdots & \ddots & \vdots \\ k_{\delta,1} & \cdots & k_{\delta,\gamma} \end{bmatrix} \in K^{\delta \times \gamma} \text{ s.t. } k_{i,j} \leftarrow \Xi.\mathsf{SKGen}(\mathsf{param}),\\
&\mathbf{P} = \Xi.\mathsf{PKGen}(\mathbf{K_I}), \quad \mathsf{SK_I} = (\mathbf{I}_\delta, \mathbf{K_I}).
\end{aligned}$$

return $(\mathsf{PK}, \mathsf{MSK}) = (\mathbf{P}, \mathsf{SK_I})$.

– $\mathsf{SK_{BA}} \leftarrow \mathsf{KeyGen}(\mathsf{SK_A}, \mathbf{B})$: In the key generation algorithm, the algorithm recognises the secret key $\mathsf{SK_A}$ for the matrix $\mathbf{A}$. If the secret key is not in the form of $\mathsf{SK_A} = (\mathbf{A}, \mathbf{K_A}) \in \mathbb{Z}_n^{m \times \delta} \times K^{m \times \gamma}$ for some $m \in \mathbb{Z}^+$, the algorithm returns $\perp$ for failure indication. The algorithm also checks the validity of the parameter $\mathbf{B} \in \mathbb{Z}_n^{m' \times m}$ for some $m' \in \mathbb{Z}^+$. If all parameters are valid and compatible, the algorithm computes and returns a secret key $\mathsf{SK_{BA}} = (\mathbf{BA}, \mathbf{BK_A}) \in \mathbb{Z}_n^{m' \times \delta} \times K^{m' \times \gamma}$ for the matrix $\mathbf{BA}$. Remark that $\mathbf{BK_A} = \mathbf{BAK_I}$.

– $C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, \mathbf{X})$: To encrypt a matrix $\mathbf{X} \in \mathbb{Z}_n^{\delta \times \gamma}$, the algorithm randomly samples a word x in the language L with a witness w such that $(x, w) \in R$. After that, the algorithm computes the core part of the ciphertext $\mathbf{C} = \mathbf{X}\xi + \Xi.\mathsf{PHash}(\mathbf{P}, x, w) \in \Pi^{\delta \times \gamma}$. The full ciphertext is $C = (x, \mathbf{C})$.

$$(x, w) \in_R R, \quad \mathbf{C} = \mathbf{X}\xi + \Xi.\mathsf{PHash}(\mathbf{P}, x, w).$$

return $C = (x, \mathbf{C})$.

– $\mathbf{Y} \leftarrow \mathsf{Decrypt}(\mathsf{SK_A}, C)$: To decrypt a ciphertext $C = (x, \mathbf{C}) \in X \times \Pi^{\delta \times \gamma}$ with a secret key $\mathsf{SK_A} = (\mathbf{A}, \mathbf{K_A}) \in \mathbb{Z}_n^{m \times \delta} \times K^{m \times \gamma}$, the algorithm computes

[1]From the key linearity and hash linearity, we have that $|K| \geq |\Pi| \geq |\Pi'| = n$, and n could be maximised by summing two or more elements derived from the diversity if those elements generate different groups.

an intermediate value $\mathbf{D} = \mathbf{AC} - \Xi.\mathsf{Hash}(\mathbf{K_A}, x) \in \Pi^{m\times\gamma}$. After that, the algorithm find the scalar of $\mathbf{D}$ with the base ξ as the final decryption result $\mathbf{Y} = \mathrm{sca}_\xi(\mathbf{D}) \in \mathbb{Z}_n^{m\times\gamma}$.

$$\mathbf{D} = \mathbf{AC} - \Xi.\mathsf{Hash}(\mathbf{K_A}, x).$$

return $\mathbf{Y} = \mathrm{sca}_\xi(\mathbf{D})$.

We show that our construction is complete by verifying the decryption algorithm. Starting from the intermediate value $\mathbf{D}$, we have

$$\begin{aligned}
\mathbf{D} &= \mathbf{AC} - \Xi.\mathsf{Hash}(\mathbf{K_A}, x) \\
&= \mathbf{A}(\mathbf{X}\xi + \Xi.\mathsf{PHash}(\mathbf{P}, x, w)) - \Xi.\mathsf{Hash}(\mathbf{K_A}, x) \\
&= \mathbf{AX}\xi + \mathbf{A}\Xi.\mathsf{PHash}(\mathbf{P}, x, w) - \Xi.\mathsf{Hash}(\mathbf{K_A}, x) \\
&= \mathbf{AX}\xi + \mathbf{A}\Xi.\mathsf{Hash}(\mathbf{K_I}, x) - \Xi.\mathsf{Hash}(\mathbf{K_A}, x) \\
&= \mathbf{AX}\xi + \Xi.\mathsf{Hash}(\mathbf{AK_I}, x) - \Xi.\mathsf{Hash}(\mathbf{AK_I}, x) \\
&= \mathbf{AX}\xi.
\end{aligned}$$

Then the completeness of our construction is verified by

$$\mathbf{Y} = \mathrm{sca}_\xi(\mathbf{D}) = \mathrm{sca}_\xi(\mathbf{AX}\xi) = \mathbf{AX}.$$

7.3.2 Security Proof

Theorem 7.1. *The proposed HFE-LT construction in Section 7.3.1 is IND-CPA secure if the SMP instance Λ associated with the underlying diverse HPS Ξ is hard.*

Proof. In this proof, we require the underlying HPS Ξ to have diversity instead of smoothness introduced by Cramer and Shoup [CS02], which is used to prove the security of an IND-CPA PKE scheme. The reason is that the smoothness only prevents the information leakage of the hashing value from the public hash keys but not from the secret hash keys. In other words, the adversary may be able to distinguish ciphertexts via secret keys obtained from the key generation algorithm KeyGen of the HFE-LT scheme. If we follow the proof in [CS02] that the hash values are replaced with random values, the adversary can recognise this game modification with overwhelming probability by running the decryption algorithm since the adversary finds that $\mathsf{Decrypt}(\mathsf{SK_A}, C) \neq \mathbf{AX}_0$ and $\mathsf{Decrypt}(\mathsf{SK_A}, C) \neq \mathbf{AX}_1$ where C is the challenge ciphertext of $\mathbf{X}_0$ or $\mathbf{X}_1$. Moreover, we are still able to use a perfectly smooth HPS since it implies diversity (see Section 6.3.1 for more details).

To prove the theorem, we show that an simulator $\mathcal{S}$ can be constructed to solve

the SMP instance $\Lambda = (X, L, W, R)$ in polynomial time with non-negligible probability if an adversary $\mathcal{A}$ can win the IND-CPA game with non-negligible probability.

Let (Λ, x^*) be the actual subset membership problem challenged to the simulator $\mathcal{S}$. The objective of the simulator $\mathcal{S}$ is to distinguish whether $x^* \in L$ or $x^* \in X \setminus L$ with x^* sampled from L or $X \setminus L$ with equal probability. Following the IND-CPA game (Fig. 3.1 defined in Section 3.2.2), the simulator $\mathcal{S}$ runs the system setup algorithm Setup to generate a key pair $(\mathsf{PK}, \mathsf{MSK}) = (\mathbf{P}, \mathsf{SK}_{\mathbf{I}})$, and passes the public key PK to the adversary $\mathcal{A}$ in the setup phase. During the pre-challenge phase, the simulator $\mathcal{S}$ invokes the key generation algorithm KeyGen with the master secret key $\mathsf{MSK} = \mathsf{SK}_{\mathbf{I}}$ to answer the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}$ directly. The restriction for the adversary $\mathcal{A}$ is that $\mathcal{A}$ can only query the secret keys for $\mathbf{A}$ such that $\mathbf{AX}_0 = \mathbf{AX}_1$ where $\mathbf{X}_0$ and $\mathbf{X}_1$ are the target message matrices output by $\mathcal{A}$ in the challenge phase.

At some point, the adversary $\mathcal{A}$ outputs two target message matrices $\mathbf{X}_0$ and $\mathbf{X}_1$, changing the game state to the challenge phase. The simulator $\mathcal{S}$ tosses a random coin $b \in_R \{0, 1\}$, and computes the target ciphertext $C^* = (x^*, \mathbf{C}^*)$ different to the encryption algorithm $\mathsf{Encrypt}$ where

$$\mathbf{C}^* = \mathbf{X}_b \xi + \Xi.\mathsf{Hash}(\mathbf{K}_{\mathbf{I}}, x^*).$$

The target ciphertext C^* is then sent to the adversary $\mathcal{A}$. In the post-challenge phase, the adversary $\mathcal{A}$ is allowed to access the oracle $\mathcal{O}_{\mathsf{KeyGen}}$ as before with the same restriction. Eventually, the adversary $\mathcal{A}$ outputs a bit b' in the guessing phase. If $b = b'$, the adversary $\mathcal{A}$ wins, and the simulator $\mathcal{S}$ outputs 1. Otherwise, the simulator $\mathcal{S}$ outputs 0 instead. Finally, the simulator $\mathcal{S}$ halts and completes the simulation.

After the simulation, we analyse the probabilities in the style of [CS02, ZMY17]. Let E_L be the event that $\mathcal{S}$ outputs 1 conditioned on $x^* \in L$, and $E_{X \setminus L}$ be the event that $\mathcal{S}$ outputs 1 conditioned on $x^* \in X \setminus L$. The advantage $\mathsf{Adv}_{\mathcal{S}}^{\mathrm{SMP}}$ of solving the subset membership problem is

$$\mathsf{Adv}_{\mathcal{S}}^{\mathrm{SMP}} \geq |\Pr[1 \leftarrow \mathcal{S} \mid x^* \in L] - \Pr[1 \leftarrow \mathcal{S} \mid x^* \in X \setminus L]| = \left|\Pr[E_L] - \Pr[E_{X \setminus L}]\right| \tag{7.1}$$

For the case of $x^* \in L$, the simulation is perfect since the algorithms $\Xi.\mathsf{Hash}$ and $\Xi.\mathsf{PHash}$ are equivalent. From Fig. 3.1, we have

$$\mathsf{Adv}_{\mathcal{A}}^{\text{IND-CPA}} = \left|\Pr[E_L] - \frac{1}{2}\right|. \tag{7.2}$$

For the case of $x^* \in X \setminus L$, we show that the hidden bit b is independent from the

adversary $\mathcal{A}$'s view that

$$\Pr[E_{X\setminus L}] = \frac{1}{2} \tag{7.3}$$

Since the element ξ is an element derived from the the diversity of the HPS Ξ, we have that there exists a $k \in K$ such that $\mathsf{PKGen}(k) = 0$ and $\Xi.\mathsf{Hash}(k, x^*) = \xi$ where k is not required to be efficiently computable. Let $\mathbf{\Gamma} = (\mathbf{X}_b - \mathbf{X}_{1-b}) \cdot k \in K^{\delta\times\gamma}$. We have

$$\begin{aligned}
\Xi.\mathsf{Hash}(\mathbf{\Gamma}, x^*) &= \Xi.\mathsf{Hash}((\mathbf{X}_b - \mathbf{X}_{1-b}) \cdot k, x^*) \\
&= (\mathbf{X}_b - \mathbf{X}_{1-b}) \cdot \Xi.\mathsf{Hash}(k, x^*) \\
&= (\mathbf{X}_b - \mathbf{X}_{1-b})\xi \\
&= \mathbf{X}_b\xi - \mathbf{X}_{1-b}\xi.
\end{aligned}$$

Based on $\Xi.\mathsf{Hash}(\mathbf{\Gamma}, x^*) = \mathbf{X}_b\xi - \mathbf{X}_{1-b}\xi$, we argue that the target ciphertext C^* is not only a valid ciphertext of the message $\mathbf{X}_b$ under the key $\mathbf{K_I}$ but also a valid ciphertext of the message $\mathbf{X}_{1-b}$ under the key $\mathbf{K}_\mathbf{I}^* = \mathbf{K_I} + \mathbf{\Gamma}$. More precisely, we have

$$\begin{aligned}
\mathbf{C}^* &= \mathbf{X}_{1-b}\xi + \Xi.\mathsf{Hash}(\mathbf{K_I} + \mathbf{\Gamma}, x^*) \\
&= \mathbf{X}_{1-b}\xi + \Xi.\mathsf{Hash}(\mathbf{K_I}, x^*) + \Xi.\mathsf{Hash}(\mathbf{\Gamma}, x^*) \\
&= \mathbf{X}_{1-b}\xi + \Xi.\mathsf{Hash}(\mathbf{K_I}, x^*) + \mathbf{X}_b\xi - \mathbf{X}_{1-b}\xi \\
&= \mathbf{X}_b\xi + \Xi.\mathsf{Hash}(\mathbf{K_I}, x^*).
\end{aligned}$$

Furthermore, we show that it is impossible for the adversary $\mathcal{A}$ to distinguish $\mathbf{K_I}$ and $\mathbf{K}_\mathbf{I}^*$ from the public key $\mathbf{P}$ or the secret keys $\mathbf{K_A}$ obtained from the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}$. Since $\Xi.\mathsf{PKGen}(k) = 0$ from the diversity, the keys $\mathbf{K_I}$ and $\mathbf{K}_\mathbf{I}^*$ have the same public key

$$\begin{aligned}
\mathbf{P} &= \Xi.\mathsf{PKGen}(\mathbf{K_I} + \mathbf{\Gamma}) \\
&= \Xi.\mathsf{PKGen}(\mathbf{K_I}) + \Xi.\mathsf{PKGen}(\mathbf{\Gamma}) \\
&= \Xi.\mathsf{PKGen}(\mathbf{K_I}) + \Xi.\mathsf{PKGen}((\mathbf{X}_b - \mathbf{X}_{1-b}) \cdot k) \\
&= \Xi.\mathsf{PKGen}(\mathbf{K_I}) + (\mathbf{X}_b - \mathbf{X}_{1-b}) \cdot \Xi.\mathsf{PKGen}(k) \\
&= \Xi.\mathsf{PKGen}(\mathbf{K_I}).
\end{aligned}$$

Since the adversary $\mathcal{A}$ is restricted that $\mathcal{A}$ can only query the secret keys for $\mathbf{A}$ such that $\mathbf{AX}_0 = \mathbf{AX}_1 \iff \mathbf{A}(\mathbf{X}_0 - \mathbf{X}_1) = 0$, the keys $\mathbf{K_I}$ and $\mathbf{K}_\mathbf{I}^*$ generate the same secret key $\mathbf{K_A}$ for such a matrix $\mathbf{A}$ for the adversary $\mathcal{A}$ as

$$\mathbf{K_A} = \mathbf{A}(\mathbf{K_I} + \mathbf{\Gamma}) = \mathbf{AK_I} + \mathbf{A\Gamma} = \mathbf{AK_I} + \mathbf{A}(\mathbf{X}_b - \mathbf{X}_{1-b}) \cdot k = \mathbf{AK_I}.$$

Hence, the hidden bit b is independent from the adversary $\mathcal{A}$'s view.

By combining Eqs. (7.1) to (7.3), we have the following inequality and complete the proof.

$$\mathsf{Adv}_{\mathcal{A}}^{\text{IND-CPA}} \leq \mathsf{Adv}_{\mathcal{S}}^{\text{SMP}}.$$

□

7.4 Extensions

In the Definition 7.1, the delegation is performed by one party with a secret key SK_f and a delegation function f' so that the delegated party can learn $f'(f(x))$. In order not to be restricted to nested hierarchy, the HFE can be extended to allow delegation performed by multiple parties. Precisely, n secret key holders with $\mathsf{SK}_{f_1}, \ldots, \mathsf{SK}_{f_n}$ can work together to generate a secret key SK_{f^*} for a function f^* defined as $f^*(x) = f'(f_1(x), \ldots, f_n(x))$ where f' is a delegation function, which takes n inputs. The extended HFE is formalised as follows.

Definition 7.3 (extended Hierarchical Functional Encryption)**.** *An extended Hierarchical Functional Encryption (eHFE) for a function class $\mathcal{F} = \{f_k : X \to Y \mid k \in K\}$ consists of the following four polynomial time algorithms:*

$$(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda), \qquad \mathsf{SK}_{f_k} \leftarrow \mathsf{KeyGen}(\mathsf{SK}_{f_{k_1}}, \ldots \mathsf{SK}_{f_{k_n}}, f'),$$
$$C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, x), \qquad y = f(x) \leftarrow \mathsf{Decrypt}(\mathsf{SK}_f, C).$$

While all other components work as in Definition 7.1, the extended key generation algorithm KeyGen *takes n secret keys for function $f_{k_i} \in \mathcal{F} : X \to Z_i \subset Y$, and a function $f' : Z_1 \times \cdots \times Z_n \to Z' \subset Y$ as inputs such that $f_k(x) \stackrel{\text{def}}{=} f'(f_{k_1}(x), \ldots, f_{k_2}(x))$ and $f_k \in \mathcal{F}$ for some $k \in K$. The algorithm* KeyGen *outputs a secret key for the function f_k. It is worth noting the number n of parameters of the algorithm* KeyGen *is not fixed by the system setup algorithm* Setup*.*

Similar to HFE, the definition of the IND-CPA security (Definition 3.2) can also be applied to eHFE with the same method to resolve the queries to the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}$. Based on Definitions 7.2 and 7.3, we derive the syntax of our extended HFE-LT.

Definition 7.4 (extended Hierarchical Functional Encryption for Linear Transformations)**.** *Let $\mathbb{R}$ be a ring, $K = \{\mathbf{A} \mid \mathbf{A} \in \mathbb{R}^{i \times \delta}, i \in \mathbb{Z}^+\}$, $X = \mathbb{R}^{\delta \times \gamma}$, and $Y = \{\mathbf{Y} \mid \mathbf{Y} \in \mathbb{R}^{i \times \gamma}, i \in \mathbb{Z}^+\}$. The universal function $f : K \times X \to Y$ is defined as $f_{\mathbf{A}} : \mathbf{X} \mapsto \mathbf{AX}$. An extended Hierarchical Functional Encryption for Linear Transformations (eHFE-LT) is an eHFE for a function class $\mathcal{F} = \{f_k : X \to Y \mid k \in K\}$,*

consisting of the following four polynomial time algorithms:

$$(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\delta, 1^\gamma), \qquad \mathsf{SK}_{\mathbf{B}} \leftarrow \mathsf{KeyGen}(\mathsf{SK}_{\mathbf{A}_1}, \ldots, \mathsf{SK}_{\mathbf{A}_n}, \mathbf{T}),$$
$$C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, \mathbf{X}), \qquad \mathbf{Y} = \mathbf{AX} \leftarrow \mathsf{Decrypt}(\mathsf{SK}_{\mathbf{A}}, C).$$

In the key generation algorithm KeyGen*, the resulted transformation matrix is calculated as*

$$\mathbf{B} = \mathbf{T}\left(\mathbf{A}_1^\top \,\middle|\, \cdots \,\middle|\, \mathbf{A}_n^\top\right)^\top.$$

The construction of our eHFE-LT scheme is exactly the same as the HFE-LT scheme in Section 7.3.1 except the key generation algorithm. The idea of constructing the new key generation algorithm is simple due to the special structure of the secret keys that we combine the keys $\mathsf{SK}_{\mathbf{A}_1}, \ldots, \mathsf{SK}_{\mathbf{A}_\ell}$ to be a new key $\mathsf{SK}_{\mathbf{A}}$ where $\mathbf{A} = \left(\mathbf{A}_1^\top \,\middle|\, \cdots \,\middle|\, \mathbf{A}_\ell^\top\right)^\top$, and then runs the original key generation algorithm $\mathsf{KeyGen}(\mathsf{SK}_{\mathbf{A}}, \mathbf{T})$ to obtain the final key $\mathsf{SK}_{\mathbf{B}}$. More precisely, we have

– $\mathsf{SK}_{\mathbf{B}} \leftarrow \mathsf{KeyGen}(\mathsf{SK}_{\mathbf{A}_1}, \ldots, \mathsf{SK}_{\mathbf{A}_\ell}, \mathbf{T})$: Given ℓ secret keys for $\mathbf{A}_i \in \mathbb{Z}_n^{m_i \times \delta}$, the algorithm checks the validity of $\mathbf{T} \in \mathbb{Z}_n^{m' \times \sum_{i=1}^{\ell} m_i}$ for some $m' \in \mathbb{Z}^+$. Then it computes

$$\mathbf{B} = \mathbf{T}\left(\mathbf{A}_1^\top \,\middle|\, \cdots \,\middle|\, \mathbf{A}_\ell^\top\right)^\top, \quad \mathbf{K}_{\mathbf{B}} = \mathbf{T}\left(\mathbf{K}_{\mathbf{A}_1}^\top \,\middle|\, \cdots \,\middle|\, \mathbf{K}_{\mathbf{A}_\ell}^\top\right)^\top.$$

return $\mathsf{SK}_{\mathbf{B}} = (\mathbf{B}, \mathbf{K}_{\mathbf{B}}) \in \mathbb{Z}_n^{m' \times \delta} \times K^{m' \times \gamma}$.

Theorem 7.2. *The proposed eHFE-LT construction in Section 7.4 is IND-CPA secure if the SMP instance Λ associated with the underlying diverse HPS Ξ is hard.*

Proof. The proof follows the same lines as in Section 7.3.2 and is omitted. □

7.5 Instantiations

In this section, we instantiate our generic HFE-LT construction from the DDH problem and from the DCR problem. Remark that the HFE-LT instantiations can be easily converted to eHFE-LT instantiations as discussed in Section 7.4. For better readability, we use multiplication in this section instead of addition in the previous sections for the group operations related to HPS. Therefore, all scalar multiplications on groups related to HPS are replaced by exponentiations.

7.5.1 From Decisional Diffie-Hellman Assumption

We recall the DDH-based HPS construction (construction 6.5 in Section 6.5.1), which has key linearity, hash linearity, and diversity with g_1 as a derived element[2].

[2]All elements in $\mathbb{G}$ are elements derived from the diversity.

Let $\xi = g_1$. Our HFE-LT instantiation of $\mathbb{R} = \mathbb{Z}_p$ works as follows.

– $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\delta, 1^\gamma)$:

$$g_1, g_2 \in \mathbb{G}, \quad \mathbf{K_I} = \begin{bmatrix} (k_{1,1,1}, k_{1,1,2}) & \cdots & (k_{1,\gamma,1}, k_{1,\gamma,2}) \\ \vdots & \ddots & \vdots \\ (k_{\delta,1,1}, k_{\delta,1,2}) & \cdots & (k_{\delta,\gamma,1}, k_{\delta,\gamma,2}) \end{bmatrix} \in_R (\mathbb{Z}_p^2)^{\delta\times\gamma},$$

$$\mathbf{P} = \begin{bmatrix} P_{1,1} & \cdots & P_{1,\gamma} \\ \vdots & \ddots & \vdots \\ P_{\delta,1} & \cdots & P_{\delta,\gamma} \end{bmatrix} \text{ s.t. } P_{i,j} = g_1^{k_{i,j,1}} g_2^{k_{i,j,2}}, \quad \mathsf{SK_I} = (\mathbf{I}_\delta, \mathbf{K_I}).$$

return $(\mathsf{PK}, \mathsf{MSK}) = ((g_1, g_2, \mathbf{P}), \mathsf{SK_I})$.

– $\mathsf{SK_{BA}} \leftarrow \mathsf{KeyGen}(\mathsf{SK_A}, \mathbf{B})$: return $\mathsf{SK_{BA}} = (\mathbf{BA}, \mathbf{BK_A})$.

– $C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, \mathbf{X})$:

$$w \in_R \mathbb{Z}_p, \quad x = (x_1, x_2) = (g_1^w, g_2^w), \quad \begin{bmatrix} X_{1,1} & \cdots & X_{1,\gamma} \\ \vdots & \ddots & \vdots \\ X_{\delta,1} & \cdots & X_{\delta,\gamma} \end{bmatrix} \leftarrow \mathbf{X} \in \mathbb{Z}_p^{\delta\times\gamma},$$

$$\mathbf{C} = \begin{bmatrix} C_{1,1} & \cdots & C_{1,\gamma} \\ \vdots & \ddots & \vdots \\ C_{\delta,1} & \cdots & C_{\delta,\gamma} \end{bmatrix} \text{ s.t. } C_{i,j} = \xi^{X_{i,j}} P_{i,j}^w = g_1^{X_{i,j}} P_{i,j}^w.$$

return $C = (x, \mathbf{C})$.

– $\mathbf{Y} \leftarrow \mathsf{Decrypt}(\mathsf{SK_A}, C)$:

$$\begin{bmatrix} (k_{1,1,1}, k_{1,1,2}) & \cdots & (k_{1,\gamma,1}, k_{1,\gamma,2}) \\ \vdots & \ddots & \vdots \\ (k_{m,1,1}, k_{m,1,2}) & \cdots & (k_{m,\gamma,1}, k_{m,\gamma,2}) \end{bmatrix} \leftarrow \mathbf{K_A} \in (\mathbb{Z}_p^2)^{m\times\gamma},$$

$$\begin{bmatrix} A_{1,1} & \cdots & A_{1,\delta} \\ \vdots & \ddots & \vdots \\ A_{m,1} & \cdots & A_{m,\delta} \end{bmatrix} \leftarrow \mathbf{A} \in \mathbb{Z}_p^{m\times\delta},$$

$$\mathbf{D} = \begin{bmatrix} D_{1,1} & \cdots & D_{1,\gamma} \\ \vdots & \ddots & \vdots \\ D_{m,1} & \cdots & D_{m,\gamma} \end{bmatrix} \text{ s.t. } D_{i,j} = \frac{\prod_{l=1}^{\delta} C_{l,j}^{A_{i,l}}}{x_1^{k_{i,j,1}} x_2^{k_{i,j,2}}}.$$

return $\mathbf{Y} = \log_\xi \mathbf{D} = \log_{g_1} \mathbf{D}$.

Remark the calculation of $\log_{g_1} \mathbf{D}$ can be done in polynomial time if the decryption space $\{\mathbf{Y}\}$ is polynomial sized.

7.5.2 From Decisional Composite Residuosity Assumption

We recall the DCR-based HPS construction (construction 6.8 in Section 6.5.2), which has key linearity, hash linearity, and diversity with a derived element $1 + N \pmod{N^2}$.

Let $\xi = 1 + N \pmod{N^2}$. Our HFE-LT instantiation of $\mathbb{R} = \mathbb{Z}_N$ works as follows[3].

– $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\delta, 1^\gamma)$:

$$\mu \in_R \mathbb{Z}^*_{N^2}, \quad g = \mu^{2N} \pmod{N^2},$$
$$\mathbf{K_I} = \begin{bmatrix} k_{1,1} & \cdots & k_{1,\gamma} \\ \vdots & \ddots & \vdots \\ k_{\delta,1} & \cdots & k_{\delta,\gamma} \end{bmatrix} \in_R \{0, \ldots, \lfloor N^2/2 \rfloor\}^{\delta \times \gamma},$$
$$\mathbf{P} = \begin{bmatrix} P_{1,1} & \cdots & P_{1,\gamma} \\ \vdots & \ddots & \vdots \\ P_{\delta,1} & \cdots & P_{\delta,\gamma} \end{bmatrix} \text{ s.t. } P_{i,j} = g^{k_{i,j}} \pmod{N^2}, \quad \mathsf{SK_I} = (\mathbf{I}_\delta, \mathbf{K_I}).$$

return $(\mathsf{PK}, \mathsf{MSK}) = ((N, g, \mathbf{P}), \mathsf{SK_I})$.

– $\mathsf{SK_{BA}} \leftarrow \mathsf{KeyGen}(\mathsf{SK_A}, \mathbf{B})$: return $\mathsf{SK_{BA}} = (\mathbf{BA}, \mathbf{BK_A})$ where $\mathbf{BK_A}$ is computed over $\mathbb{Z}$.

– $C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, \mathbf{X})$:

$$w \in_R \{0, \ldots, \lfloor N/4 \rfloor\}, \quad x = g^w \pmod{N^2},$$
$$\begin{bmatrix} X_{1,1} & \cdots & X_{1,\gamma} \\ \vdots & \ddots & \vdots \\ X_{\delta,1} & \cdots & X_{\delta,\gamma} \end{bmatrix} \leftarrow \mathbf{X} \in \mathbb{Z}_N^{\delta \times \gamma},$$
$$\mathbf{C} = \begin{bmatrix} C_{1,1} & \cdots & C_{1,\gamma} \\ \vdots & \ddots & \vdots \\ C_{\delta,1} & \cdots & C_{\delta,\gamma} \end{bmatrix} \text{ s.t. } C_{i,j} = \xi^{X_{i,j}} P_{i,j}^w = (1 + X_{i,j}N) P_{i,j}^w \pmod{N^2}.$$

return $C = (x, \mathbf{C})$.

[3]We do not fully use the key space (i.e. $|K| = \lfloor N^2/2 \rfloor > N$).

– $\mathbf{Y} \leftarrow \mathsf{Decrypt}(\mathsf{SK}_{\mathbf{A}}, C)$:

$$\begin{bmatrix} A_{1,1} & \cdots & A_{1,\delta} \\ \vdots & \ddots & \vdots \\ A_{m,1} & \cdots & A_{m,\delta} \end{bmatrix} \leftarrow \mathbf{A} \in \mathbb{Z}_N^{m\times\delta}, \quad \begin{bmatrix} k_{1,1} & \cdots & k_{1,\gamma} \\ \vdots & \ddots & \vdots \\ k_{m,1} & \cdots & k_{m,\gamma} \end{bmatrix} \leftarrow \mathbf{K}_{\mathbf{A}} \in \mathbb{Z}^{m\times\gamma},$$

$$\mathbf{D} = \begin{bmatrix} D_{1,1} & \cdots & D_{1,\gamma} \\ \vdots & \ddots & \vdots \\ D_{m,1} & \cdots & D_{m,\gamma} \end{bmatrix} \text{ s.t. } D_{i,j} = \frac{\prod_{l=1}^{\delta} C_{l,j}^{A_{i,l}}}{x^{k_{i,j}}} \pmod{N^2},$$

$$\mathbf{Y} = \log_{\xi} \mathbf{D} = \begin{bmatrix} Y_{1,1} & \cdots & Y_{1,\gamma} \\ \vdots & \ddots & \vdots \\ Y_{m,1} & \cdots & Y_{m,\gamma} \end{bmatrix} \text{ s.t. } Y_{i,j} = \frac{D_{i,j} - 1 \mod N^2}{N}.$$

return $\mathbf{Y}$.

7.6 Chapter Summary

In this chapter, we revisited and simplified the definition of HFE, and further extended it to eHFE. We derived the notion of HFE-LT from HFE (and eHFE-LT from eHFE), allowing matrix product evaluation. Furthermore, we proposed a generic construction of HFE-LT (and eHFE-LT) with IND-CPA security in the standard model from HPS with key and hash linearity, and diversity. To illustrate that our scheme is practical, we presented two concrete HFE-LT instantiations from the DDH and DCR assumptions.

This chapter proposed a practical HFE construction for matrix product evaluation. It is still an open problem whether we could build a practical HFE for other functionalities and we leave it as our future work.

Part IV

Conclusion and Future Work

Chapter 8

Thesis Conclusion

In this chapter, we summarise the work presented in this thesis and present some future research directions.

8.1 Conclusion

In this thesis, we reviewed the notion of functional encryption, which allows decryption of partial information and captures a large class of functionalities. Particularly, we presented practical functional encryption schemes for two specific functionalities: data search and data sharing.

For cryptosystems that can be applied to data search, we introduced a new primitive named Threshold Broadcast Encryption with Keyword Search (TBEKS) and a generic conversion from Linear Encryption Template (LET) to Linear Encryption with Keyword Search (LEKS) that can be applied to various Public Key Encryption schemes. More specifically, we formally defined TBEKS and its security model named Indistinguishability in the threshold setting under Chosen Keyword Attacks (IND-T-CKA) and proposed the first TBEKS scheme with IND-T-CKA security under the Decisional Bilinear Diffie-Hellman (DBDH) assumption. For LEKS, we formalised the definition and its security models: Indistinguishability under Adaptive Chosen Keyword Attacks (IND-CKA) and Indistinguishability under Selective-ID Adaptive Chosen Keyword Attacks (IND-sCKA). We proposed a Public-key Encryption with Keyword Search (PEKS) scheme from a Public Key Encryption (PKE) scheme via our LEKS conversion. Moreover, we proposed the first secure Key-Policy Attribute-Based Encryption with Keyword Search (KP-ABKS) scheme converted from a Key-Policy Attribute-Based Encryption (KP-ABE) scheme. With our KP-ABKS, we were able to do privacy-preserved keyword search with fine-grained access control in the cloud. Our PEKS and KP-ABKS schemes were proven secure based on the Decisional Bilinear (P, f)-Diffie-Hellman ((P, f)-DBDH) problem family and Decisional ℓ-Combined Bilinear Diffie-Hellman (ℓ-DCBDH) problem, which we defined and proved computationally hard in the generic group model.

For cryptosystems that can be applied to data sharing, we proposed a Func-

tional Encryption for Inner Products (FE-IP) scheme and a Hierarchical Functional Encryption for Linear Transformations (HFE-LT) scheme, which allow the user to learn only partial information about the encrypted data from decryption. The FE-IP we proposed is proven secure under an enhanced security model named Indistinguishability under adaptive Chosen Ciphertext Attacks (IND-CCA). In order to construct an IND-CCA secure FE-IP, we extended the Hash Proof System (HPS) with new properties, which is of independent interest. Using the extended HPS, we constructed our HFE-LT based on a refined definition of Hierarchical Functional Encryption (HFE). Notably in our proposed HFE-LT, the decryption ability is levelled and a lower level secret key can be generated by the secret key holder of the upper level. In addition, we extended HFE and HFE-LT to extended Hierarchical Functional Encryption (eHFE) and extended Hierarchical Functional Encryption for Linear Transformations (eHFE-LT) where the decryption ability is able to be derived from multiple parties instead of a single one. Finally, we proposed instantiations of FE-IP and HFE-LT from Decisional Diffie-Hellman (DDH) and Decisional Composite Residuosity (DCR) assumptions.

Besides, we proposed an encoding technique and improved the efficiency of the inequality test in Attribute-Based Encryption (ABE).

8.2 Future Work

In data search, we proposed a semi-generic framework that converts an encryption scheme to a keyword search scheme. An interesting future work is to find out more LET-compatible encryption schemes in order to enable new keyword search paradigms based on various access control mechanisms. Furthermore, we did not consider the Keyword Guessing Attack (KGA) for our schemes. From the view of Functional Encryption for Equality Tests (FE-ET), it is impossible to prevent such an attack in the public key setting. However, it is still an interesting topic to find a different method to do privacy-preserved keyword search with resistance to KGA.

In data sharing, we focused on the practical functionalities of inner products and matrix products. Finding new practical and meaningful functionalities and constructing the corresponding Functional Encryption (FE) schemes are perhaps two of the most interesting open problems in this field.

Bibliography

[ABC+08] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. *Journal of Cryptology*, 21(3):350–391, 2008.

[ABCP16] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Better security for functional encryption for inner product evaluations. Cryptology ePrint Archive, Report 2016/011, 2016. `http://eprint.iacr.org/`.

[ABDCP15] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In Jonathan Katz, editor, *Public-Key Cryptography – PKC 2015: 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 – April 1, 2015, Proceedings*, pages 733–751, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. `http://eprint.iacr.org/2013/689`.

[AFI06] Nuttapong Attrapadung, Jun Furukawa, and Hideki Imai. Forward-secure and searchable broadcast encryption with short ciphertexts and private keys. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 161–177. Springer Berlin Heidelberg, 2006.

[ALdP11] Nuttapong Attrapadung, Benoît Libert, and Elie de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio

Nicolosi, editors, *Public Key Cryptography – PKC 2011*, volume 6571 of *Lecture Notes in Computer Science*, pages 90–108. Springer Berlin Heidelberg, 2011.

[ALS16] Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, pages 333–362, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[ARW16] Michel Abdalla, Mariana Raykova, and Hoeteck Wee. Multi-input inner-product functional encryption from pairings. Cryptology ePrint Archive, Report 2016/425, 2016. `http://eprint.iacr.org/2016/425`.

[BB07] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, 2007.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. Cryptology ePrint Archive, Report 2005/015, 2005. `http://eprint.iacr.org/2005/015`.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matt Franklin, editor, *Advances in Cryptology – CRYPTO 2004: 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004. Proceedings*, pages 41–55, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[BCP03] Emmanuel Bresson, Dario Catalano, and David Pointcheval. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In Chi-Sung Laih, editor, *Advances in Cryptology - ASIACRYPT 2003: 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 – December 4, 2003. Proceedings*, pages 37–54, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[BDCOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and JanL. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer Berlin Heidelberg, 2004.

[BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer Berlin Heidelberg, 2001.

[BJK15] Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In Tetsu Iwata and Hee Jung Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015: 21st International Conference on the Theory and Application of Cryptology and Information Security,Auckland, New Zealand, November 29 – December 3, 2015, Proceedings, Part I*, pages 470–491, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, pages 62–73, New York, NY, USA, 1993. ACM.

[BS15] Zvika Brakerski and Gil Segev. Hierarchical functional encryption. Cryptology ePrint Archive, Report 2015/1011, 2015. `http://eprint.iacr.org/2015/1011`.

[BSNS06] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. On the integration of public key data encryption and public key encryption with keyword search. In SokratisK. Katsikas, Javier López, Michael Backes, Stefanos Gritzalis, and Bart Preneel, editors, *Information Security*, volume 4176 of *Lecture Notes in Computer Science*, pages 217–232. Springer Berlin Heidelberg, 2006.

[BSW07] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *Security and Privacy, 2007. SP '07. IEEE Symposium on*, pages 321–334, May 2007.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *Theory of Cryptography*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273. Springer Berlin Heidelberg, 2011.

[BW06] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 290–307. Springer Berlin Heidelberg, 2006.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In SalilP. Vadhan, editor, *Theory of Cryptography*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554. Springer Berlin Heidelberg, 2007.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 209–218, New York, NY, USA, 1998. ACM.

[CGJS15] Nishanth Chandran, Vipul Goyal, Aayush Jain, and Amit Sahai. Functional encryption: Decentralised and delegatable. Cryptology ePrint Archive, Report 2015/1017, 2015. `http://eprint.iacr.org/2015/1017`.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology — CRYPTO '98: 18th Annual International Cryptology Conference Santa Barbara, California, USA August 23–27, 1998 Proceedings*, pages 13–25, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002: International Conference on the Theory and Applications of Cryptographic Techniques Amsterdam, The Netherlands, April 28 – May 2, 2002 Proceedings*, pages 45–64, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. Full version at `http://eprint.iacr.org/2001/085`.

[CW79] J.Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143 – 154, 1979.

[Dam88] Ivan Bjerre Damgård. Collision free hash functions and public key signature schemes. In David Chaum and Wyn L. Price, editors, *Advances in Cryptology — EUROCRYPT' 87: Workshop on the Theory and Application of Cryptographic Techniques Amsterdam, The Netherlands, April 13–15, 1987 Proceedings*, pages 203–216, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[DH76] W. Diffie and M.E. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, 1976.

[DHMR07] Vanesa Daza, Javier Herranz, Paz Morillo, and Carla Ràfols. Cca2-secure threshold broadcast encryption with shorter ciphertexts. In Willy Susilo, JosephK. Liu, and Yi Mu, editors, *Provable Security*, volume 4784 of *Lecture Notes in Computer Science*, pages 35–50. Springer Berlin Heidelberg, 2007.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In George Robert Blakley and David Chaum, editors, *Advances in Cryptology: Proceedings of CRYPTO 84*, pages 10–18, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.

[FN94] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *Advances in Cryptology — CRYPTO' 93: 13th Annual International Cryptology Conference Santa Barbara, California, USA August 22–26, 1993 Proceedings*, pages 480–491, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[Gen06] Craig Gentry. Practical identity-based encryption without random oracles. In *Proceedings of the 24th Annual International Conference on The Theory and Applications of Cryptographic Techniques*, EUROCRYPT'06, pages 445–464, Berlin, Heidelberg, 2006. Springer-Verlag.

[GGG+14] Shafi Goldwasser, S.Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In PhongQ. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 578–602. Springer Berlin Heidelberg, 2014.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. Cryptology ePrint Archive, Report 2013/451, 2013. `http://eprint.iacr.org/2013/451`.

[GKP+13] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 555–564, New York, NY, USA, 2013. ACM.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270 – 299, 1984.

[Gol01] Oded Goldreich. *Foundations of cryptography.* Cambridge University Press, New York, 2001.

[Gol08] Oded Goldreich. *Computational Complexity: A Conceptual Perspective.* Cambridge University Press, New York, NY, USA, 1 edition, 2008.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 89–98, New York, NY, USA, 2006. ACM.

[HL07] YongHo Hwang and PilJoong Lee. Public key encryption with conjunctive keyword search and its extension to a multi-user system. In Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto, editors, *Pairing-Based Cryptography – Pairing 2007*, volume 4575 of *Lecture Notes in Computer Science*, pages 2–22. Springer Berlin Heidelberg, 2007.

[HW14] Susan Hohenberger and Brent Waters. Online/offline attribute-based encryption. In Hugo Krawczyk, editor, *Public-Key Cryptography – PKC 2014*, volume 8383 of *Lecture Notes in Computer Science*, pages 293–310. Springer Berlin Heidelberg, 2014.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162. Springer Berlin Heidelberg, 2008.

[LCW13a] Zhen Liu, Zhenfu Cao, and D.S. Wong. White-box traceable ciphertext-policy attribute-based encryption supporting any monotone access structures. *Information Forensics and Security, IEEE Transactions on*, 8(1):76–88, Jan 2013.

[LCW13b] Zhen Liu, Zhenfu Cao, and Duncan S. Wong. Blackbox traceable cp-abe: How to catch people leaking their keys by selling decryption devices on ebay. In *Proceedings of the 2013 ACM SIGSAC Conference on*

Computer & Communications Security, CCS '13, pages 475–486, New York, NY, USA, 2013. ACM.

[LOS+10] Allison Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 62–91. Springer Berlin Heidelberg, 2010.

[Mao03] Wenbo Mao. *Modern Cryptography: Theory and Practice*. Prentice Hall Professional Technical Reference, 2003.

[MG11] Peter Mell and Timothy Grance. The nist definition of cloud computing. Technical report, National Institue of Standards and Technology, 2011.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. `http://eprint.iacr.org/`.

[OT11] Tatsuaki Okamoto and Katsuyuki Takashima. Some key techniques on pairing vector spaces. In Abderrahmane Nitaj and David Pointcheval, editors, *Progress in Cryptology – AFRICACRYPT 2011*, volume 6737 of *Lecture Notes in Computer Science*, pages 380–382. Springer Berlin Heidelberg, 2011.

[OT12] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure unbounded inner-product and attribute-based encryption. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 349–366. Springer Berlin Heidelberg, 2012.

[OT15] Tatsuaki Okamoto and Katsuyuki Takashima. Achieving short ciphertexts or short secret-keys for adaptively secure general inner-product encryption. *Designs, Codes and Cryptography*, 77(2):725–771, 2015.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings*, pages 223–238, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[PTMW06] Matthew Pirretti, Patrick Traynor, Patrick McDaniel, and Brent Waters. Secure attribute-based systems. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 99–112, New York, NY, USA, 2006. ACM.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 84–93, New York, NY, USA, 2005. ACM.

[RS92] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91: Proceedings*, pages 433–444, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[SBC+07] Elaine Shi, John Bethencourt, T-H. Hubert Chan, Dawn Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, SP '07, pages 350–364, Washington, DC, USA, 2007. IEEE Computer Society.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In GeorgeRobert Blakley and David Chaum, editors, *Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer Berlin Heidelberg, 1985.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97: International Conference on the Theory and Application of Cryptographic Techniques Konstanz, Germany, May 11–15, 1997 Proceedings*, pages 256–266, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[Sia12] A. Siad. Anonymous identity-based encryption with distributed private-key generator and searchable encryption. In *New Technologies,*

Mobility and Security (NTMS), 2012 5th International Conference on, pages 1–8, May 2012.

[Sip96] Michael Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, 1st edition, 1996.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer Berlin Heidelberg, 2005.

[SWP00] Dawn Xiaoding Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *Security and Privacy, 2000. S P 2000. Proceedings. 2000 IEEE Symposium on*, pages 44–55, 2000.

[SYL+14] W. Sun, S. Yu, W. Lou, T. Hou, and H. Li. Protecting your right: Verifiable attribute-based keyword search with fine-grainedowner-enforced search authorization in the cloud. *Parallel and Distributed Systems, IEEE Transactions on*, PP(99):1–1, 2014.

[TCL14] Fu-Kuo Tseng, Rong-Jaye Chen, and Bao-ShuhPaul Lin. Towards symmetric functional encryption for regular languages with predicate privacy. In Maki Yoshida and Koichi Mouri, editors, *Advances in Information and Computer Security*, volume 8639 of *Lecture Notes in Computer Science*, pages 266–275. Springer International Publishing, 2014.

[Wat12] Brent Waters. Functional encryption for regular languages. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 218–235. Springer Berlin Heidelberg, 2012.

[Wat13] Brent Waters. Functional encryption: Origins and recent developments. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public-Key Cryptography – PKC 2013*, volume 7778 of *Lecture Notes in Computer Science*, pages 51–54. Springer Berlin Heidelberg, 2013.

[WC81] Mark N. Wegman and J.Lawrence Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22(3):265 – 279, 1981.

[WWP08] Peishun Wang, Huaxiong Wang, and Josef Pieprzyk. Threshold privacy preserving keyword searches. In Viliam Geffert, Juhani Karhumäki,

Alberto Bertoni, Bart Preneel, Pavol Návrat, and Mária Bieliková, editors, *SOFSEM 2008: Theory and Practice of Computer Science*, volume 4910 of *Lecture Notes in Computer Science*, pages 646–658. Springer Berlin Heidelberg, 2008.

[XJWW13] Peng Xu, Hai Jin, Qianhong Wu, and Wei Wang. Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack. *Computers, IEEE Transactions on*, 62(11):2266–2277, Nov 2013.

[ZMY16] Shiwei Zhang, Yi Mu, and Guomin Yang. Threshold broadcast encryption with keyword search. In Dongdai Lin, XiaoFeng Wang, and Moti Yung, editors, *Information Security and Cryptology: 11th International Conference, Inscrypt 2015, Beijing, China, November 1-3, 2015, Revised Selected Papers*, pages 322–337, Cham, 2016. Springer International Publishing.

[ZMY17] Shiwei Zhang, Yi Mu, and Guomin Yang. Achieving ind-cca security for functional encryption for inner products. In Kefei Chen, Dongdai Lin, and Moti Yung, editors, *Information Security and Cryptology: 12th International Conference, Inscrypt 2016, Beijing, China, November 4-6, 2016, Revised Selected Papers*, pages 119–139, Cham, 2017. Springer International Publishing.

[ZXA14] Qingji Zheng, Shouhuai Xu, and G. Ateniese. Vabks: Verifiable attribute-based keyword search over outsourced encrypted data. In *INFOCOM, 2014 Proceedings IEEE*, pages 522–530, April 2014.

[ZYM16] Shiwei Zhang, Guomin Yang, and Yi Mu. Linear encryption with keyword search. In Joseph K. Liu and Ron Steinfeld, editors, *Information Security and Privacy: 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part II*, pages 187–203, Cham, 2016. Springer International Publishing.

Appendix A

Breaking Sun et al.'s Scheme

Theorem A.1. *Sun et al.'s scheme [SYL+14] is not IND-sCP-CKA secure.*

Proof. We break the scheme by showing that an adversary $\mathcal{A}$ wins the IND-sCP-CKA game with a challenger $\mathcal{B}$ for an overwhelming advantage. For simplicity, the version control (for key revocation) part is discarded. In other words, we focus on version 1.

At the initial state, adversary $\mathcal{A}$ randomly selects a challenge access policy GT and send to the challenger $\mathcal{B}$. After that, $\mathcal{B}$ publishes public keys $\mathsf{PK} = (e, g, Y, T_1, \ldots, T_{3n})$, where $Y = e(g,g)^u$, $T_k = g^{t_k}$. Then the adversary $\mathcal{A}$ skips phase 1 and submits two randomly chosen keywords w_0, w_1 to the challenger $\mathcal{B}$. At the challenge state, the adversary $\mathcal{A}$ receives the challenge index $D_\mu = (ver, \mathsf{GT}, \hat{D}, \tilde{D}, \{D_i\}_{i\in\mathcal{N}}) \leftarrow \mathsf{EncIndex}(\mathsf{PK}, \mathsf{GT}, w_\mu)$, where $\hat{D} = g^s$, $\tilde{D} = Y^s$ and $D_i = T_i^s$ or $T_{i'}^{\frac{s}{H(w)}}$ for some public fixed position i'. As the same as in phase 1, the adversary $\mathcal{A}$ skips phase 2. Then the adversary $\mathcal{A}$ simply test the following statement.

$$e(\hat{D}, T_{i'}^{\frac{1}{H(w_{\mu'})}}) = e(g, D_{i'})$$

The adversary $\mathcal{A}$ gets either $\mu' = 0$ or $\mu' = 1$ that the above statement is true, and outputs μ' as the guess. Since

$$e(\hat{D}, T_{i'}^{\frac{1}{H(w_\mu)}}) = e(g^s, T_{i'}^{\frac{1}{H(w_\mu)}}) = e(g, T_{i'}^{\frac{s}{H(w_\mu)}}) = e(g, D_{i'}),$$

The probability for the adversary $\mathcal{A}$ winning the game is $\Pr[\mu = \mu'] = 1$. □

In other words, a malicious party can always distinguish keywords from ciphertexts in Sun et al.'s scheme.